Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.17.2.2: FIWARE GE Open Specifications (Security Chapter)

Project acronym: FI-Core
Project full title: Future Internet Core
Contract No.: 632893
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform
Project Document Number: ICT-2013-FI-632893-17-D.17.2.2
Project Document Date: 26.09.2016
Deliverable Type and Security: Public
Author: P. Bisson (Thales Services)
Contributors: Alvaro Alonso (UPM), Joaquin Salvachua (UPM), Cyril Dangerville (Thales Services)

1 Introduction

1.1 Executive Summary

This document describes the Generic Enablers in the Security chapter, their basic functionality and their interactions. These Generic Enablers form the core business framework of the FIWARE platform by supporting the business functionality for commercializing services.
The functionality of the framework is illustrated with several abstract use case diagrams, which show how the individual GE can be used to construct a domain-specific application environment and system architecture.
Each GE Open Specification is first described at a generic level, elaborating on the functional and non-functional properties. Then it is supplemented by a number of specifications according to the interface protocols, API and data formats that are delivered in separate individual documents, one per GE.
This document has the available Open Specifications that have been created in FIWARE as a result of the work in Release 5 of the platform.
There is a major shift in the approach of the Open Specifications in FIWARE. Whereas in the initial Releases (from Release 1 to Release 3) the Open Specification APIs were published on the wiki, from Release 4 onwards the information will be created and published in a more modern and manageable format using auxiliary tools such as apiary and github.
This document is accompanied by a set of annexes contained in separate documents, each one providing the detailed Open Specification API of each GE.

1.2 About This Document

FIWARE GE Open Specifications describe the open specifications linked to Generic Enablers GEs of the FIWARE platform (and their corresponding components) being developed in one particular chapter.
GE Open Specifications contain relevant information for users of FIWARE to consume related GE implementations and/or to build compliant products, which can work as alternative implementations of GEs developed in FIWARE. The later may even replace a GE implementation developed in FIWARE within a particular FIWARE instance. GE Open Specifications typically include, but not necessarily are limited to, information such as:

- Description of the scope, behaviour and intended use of the GE
- Terminology, definitions and abbreviations to clarify the meanings of the specification
- Legal information with the terms of use
- The Architecture document is generally included as is for the sake of completeness

- Signature and behaviour of operations linked to APIs (Application Programming Interfaces) that the GE should export. Signature may be specified in a particular language binding or through a RESTful interface described using API Blueprint format as per https://github.com/apiaryio/api-blueprint/blob/master/API%20Blueprint%20Specification.md.

- Description of protocols that support interoperability with other GE or third party products

- Description of non-functional features

1.3 Intended Audience

The document targets interested parties in architecture and API design, implementation and usage of FIWARE Generic Enablers from the FIWARE platform.

1.4 Structure of this Document

The document is generated out of a set of documents provided in the public FIWARE wiki. For the current version of the documents, please visit the public wiki at http://wiki.fiware.org/

The present document has been created from the wiki using automated tools and part of the links may not work. You may occasionally find oddities in the text format that side effects of the process but they do not deter the quality of the technical contents.

1.5 Keyword list

FIWARE, FI-Core, Acceleration Programme, Accelerators, PPP, Architecture Board, Steering Board, Roadmap, Reference Architecture, Generic Enabler, Open Specifications, I2ND, Cloud, IoT, Data/Media and Context Management, Applications/Services and Data Delivery, Delivery Framework, Security, Advanced Middleware, Interfaces to Networks and Robotics, Communities, Tools , Sustainability Support Tools, ICT, Internet, Apiary, Github, Latin American Platform.

1.6 Changes History

Release	Major changes description	Date	Editor
v1	Insert consolidated content from Thales & UPM	2016-09-26	Thales Services

1.7 Table of Contents

1.8 Quick Reference Table

This table contains a summary of the basic links to the detailed API on our public resources

- **Open Specification:** link to the Open Specification as included in http://wiki.fiware.org/Summary of FIWARE Open Specifications (in principle, it is the same one as in the "Structure of this Document" section.
- **API definition source:** link to the API Blueprint markdown (apib file) in GitHub. If the GE does not have a REST interface, link to alternative source if applicable.

- **API Specification Document(HTML version)**: link to the rendered definition and published as HTML in GitHub (output of the internal automated tool FABRE or alternatively Apiary output)

- **Apiary project** (optional): link to the API site in apiary.io

DETAILED OPEN SPECIFICATIONS SUMMARY TABLE		
Identity Management	Open Specification	http://wiki.fiware.org/FIWARE.OpenSpecification.Security.IdentityManagement
	API definition source (or alternative source)	https://github.com/ging/fiware-m/blob/master/extras/keyrock.apib
	API Specification Document (HTML version)	http://ging.github.io/fiware-idm/api-spec/v3
	Apiary project	http://docs.keyrock.apiary.io/
PEP Proxy	Open Specification	http://wiki.fiware.org/FIWARE.OpenSpecification.Security.PEPProxy
	API definition source (or alternative source)	N/A
	API Specification Document (HTML version)	http://wiki.fiware.org/FIWARE.OpenSpecification.Security.PEPProxy.Open_API_Specification
	Apiary project	N/A
PDP	Open Specification	http://wiki.fiware.org/FIWARE.OpenSpecification.Security.AuthorizationPDP
	API definition source (or alternative source)	https://github.com/authzforce/fiware/blob/master/apiary.apib
	API Specification Document (HTML version)	http://authzforce.github.io/fiware/authorization-pdp-api-spec/5.2/
	Apiary project	http://docs.authorizationpdp.apiary.io/

2 FIWARE.OpenSpecification.Security.IdentityManagement

Name	FIWARE.OpenSpecification.Security.IdentityManagement
Chapter	Security,
Catalogue-Link to Implementation	Keyrock
Owner	UPM , Alvaro Alonso

2.1 Preface

Within this document you find a self-contained open specification of a FIWARE generic enabler, please consult as well the FIWARE Product Vision, the website on http://www.fiware.org and similar pages in order to understand the complete context of the FIWARE platform.

2.2 Copyright

2.3 Legal Notice

2.4 Overview

On the one hand, the ever-growing tsunami of today's shore-bound technologies can often overwhelm the user, significantly affecting his daily life. On a daily basis, he/she is forced to depend on his technological competence. The smooth running of his affairs depends on the user's ability to handle a whole raft of often transient technologies. On account of very intensive, at times forced usage of the Internet and diverse services, the user encounters the need to transfer his "network-duties" to the networks as much as possible.

In other words, he/she seeks to find a convenient problem solver, which will allow him/her to cope easily and securely with services. Thus, the need arises for a clever composed Identity Management system, which will address the users' requirements.

Identity Management (IdM) encompasses some aspects involved with users' access to networks, services and applications, including secure and private authentication from users to devices,

networks and services, user profile and authorization management, Single Sign-On (SSO) to service domains and Identity Federation towards applications.
An IdM system aims to undertake the complex task of handling the various technologies in the aforementioned security domains, and provide user-friendly technologies, putting the end user and his needs squarely at the centre of the architecture (user-centric approach) whilst protecting his/her privacy.
On the other hand, more and more companies, especially small and medium enterprises are externalizing the identity management part of their applications – mostly web applications - to cloud services, so called IDaaS (Identity management as a Service). Indeed, they prefer to buy (or use for free) such IDaaS solutions rather than develop technology and/or build the infrastructure for something – identity and access management – they have no expertise in.
The Identity Management Enabler itself aims at providing IDaaS and therefore must comply with the requirements of an actual cloud-ready service, such as multi-tenancy, scalability, standard-compliant protocols and APIs, etc. As a result, instead of developing and operating the user and profile management by themselves, developers can have it hosted in the Cloud (e.g. FIWARE Lab) as a specific tenant of the IdM GE, and it will be delivered on demand.

2.5 Target usage

This enabler provides authentication, basic authorization and security assertions (such as user attributes) as a service to relying parties. The relying parties are typically service providers that provide easy and secure access to their services to users/IoT/other services for instance by means of SSO and that rely on (personal user) attributes (e.g. preferences, location, home address, etc.). The users need easy access (SSO) to the growing number of services, and many of them also prefer their personal/identity attributes to be maintained by a trusted party that also protects the users' privacy.
The Identity Management Generic Enabler can be used by such a trusted party, which we also call an identity provider (for SSO) and attribute broker. The Identity Management GE is a core Security GE that provides services to its relying parties via open protocols such as OAuth and OASIS SAML v2.0 (Security Assertion Markup Language). Motivated by IoT, the enabler also covers new user attributes such as *things* – IoT components (e.g. sensors) - owned or used by the user, as well as managing the identity of the *things* themselves (attributes, current users, location, usage history, etc.).
Furthermore, the authentication feature of the enabler also covers the authentication of *things* to users, services or other things as relying parties; and the authentication of users and services to other things as relying parties. It also supports user SSO across multiple things.

2.6 Basic Concepts

We give you short definitions of key terms in the context of the Security Chapter in the Security glossary. We now give you the extended versions of the definitions to improve your understanding of the field before we tackle the next concepts:

- **(Digital) Identity:** The term identity and its meaning have been discussed controversially in the *identity community* for many years. Until now, there is no commonly agreed definition of that notion. However, for the purpose of this

document, the Identity Gang's definition suits well (cf. http://wiki.idcommons.net/Digital_Identity):

- A digital identity is "*a digital representation of a set of Claims made by one party about itself or another digital subject.*"
- A digital identity is just one set of claims about a digital subject. For any given digital subject there will typically be many digital identities.
- A digital identity can be created on the fly when a particular identity transaction is desired or persistent in a data store to provide a representation that can be referenced.
- A digital identity may contain claims made by multiple claimants.
- A digital identity may be signed by a digital identity provider to provide assurance to a relying party.

- **(Entity) Authentication**: A simple definition is given by RFC 3588: *the act of verifying the identity of an entity (subject)*. In his book *Trust In Cyberspace* [1], Fred Schneider adds the concept *level of confidence* to this definition: *process of confirming a system entity's asserted identity with a specified, or understood, level of confidence.* This definition holds all necessary parts to examine authentication in broad sense. First of all, it does not narrow the authentication to human users, but refers to a generic *system entity*.
- **Credential**: A set of data presented as evidence of a claimed **Identity** and/or entitlements, typically for **authentication** purposes.
- **(Digital) Identity Provider**: *An identity provider is an entity that acts as an authentication service to end requestors and as data origin authentication service to service providers […]. Identity providers are trusted (logical) 3rd parties which need to be trusted both by the requestor […] and the service provider which may grant access to valuable resources and information based upon the integrity of the identity information provided by the identity provider.* (WS-Federation specification). The IdM GE can play the role of Identity Provider. It is actually one of its most important roles in FIWARE.
- **Relying party (RP):** An entity that relies on an identity representation or claim (e.g. security token or assertion) issued by a requesting/asserting entity (e.g. **Identity Provider** such as the Identity Management GE) within some request context, typically for **authentication** purposes.
- **Single Sign-On** : *From a Principal's [user's or application's] perspective, single sign-on encompasses the capability to authenticate with some system entity—[…] an **Identity Provider** - and have that authentication honored by other system entities, [termed] Service Providers [aka **Relying Parties** in this context] […]. Note that upon authenticating with an Identity Provider, the Identity Provider typically establishes and maintains some notion of local session state between itself and the Principal's user agent. Service Providers may also maintain their own distinct local session state with a Principal's user agent.* (Liberty Alliance Project).
- **Federated Identity Management (aka Federated Identity):** Identity management capabilities that allow the users of one enterprise environment (security domain, aka *realm)* to access the services of another without registering them in the user registries of the other. The WS-Federation specification gives a more formal definition: *A federation is a collection of realms that have established a producer-*

consumer relationship whereby one realm can provide authorized access to a resource it manages based on an identity, and possibly associated attributes, that are asserted in another realm. Federation requires trust such that a relying party can make a well-informed access control decision based on the credibility of identity and attribute data that is vouched for by another realm. A practical application of this is *federated SSO* (Single Sign-On): This allows users registered in one IdM X to authenticate to another IdM Y without being previously registered in the IdM Y, and, as result, to authenticate to all service providers trusting IdM Y. But Federated identity goes beyond that. For instance, it also deals with cross-domain (e.g. between IdM X's domain and Y's domain) user account provisioning (just-in-time provisioning), exchange and mapping of user attributes, cross-domain authorization, etc.

- **Authorization** : Term used interchangeably for two close but distinct meanings depending on the context:
    - **Authorization management:** Process of assigning permissions to entities. Also known as user-permission assignment. In the context of FIWARE, the IdM GE typically handles that part; therefore the term is used with that meaning in the context of the IdM, unless told otherwise.
    - **Authorization enforcement:** Process of determining whether an entity should be allowed to do something whenever the entity requests access, followed by the actual granting of access or not. In the context of FIWARE, the Authorization PDP GE determines whether access should be allowed, and based on this PDP's decision, the PEP Proxy GE actually grants or denies the requested access to the requesting entity. See the *Authorization PDP GE (§3.2)* and *PEP Proxy GE (§ 3.3)* sections for more information.

The next sections describe the higher-level concepts supporting the aforementioned features.

2.6.1 User Life-Cycle Management

The IdM offers tools for administrators to support the handling of user life-cycle functions. It reduces the effort for account creation and management, as it supports the enforcement of policies and procedures for user identity lifecycle management: user registration, user profile update, user profile removal. Administrators can quickly configure customized pages for the inclusion of different authentication providers, registration of tenant applications with access to user profile data and the handling of error notifications. For end users, the IdM provides a convenient solution for registering with applications since it gives them a means to re-use attributes like address, email or others, thus allowing an easy and convenient management of profile information. Users and administrators can rely on standardized solutions to allow user self-service features like:

- User registration/unregistration and login/logout,
- Checks for password strength,
- Password reset or renewal procedures or
- Secured storage of user data.

2.6.2 Flexible Authentication Providers

In addition to providing a native login, the Identity Provider (IdP) supports the integration of multiple 3rd party authentication providers. Foremost, it supports in a first step the configuration of preferred identity providers to lower entry barriers for a native user registration to administrators and on user side to link a preferred 3rd party IdP as alternative authentication provider to a native account. For instance, using the SAML protocol, you can trust external SAML-compliant IdPs to authenticate the users. Once the user is authenticated by a trusted external IdP, i.e. he was issued a valid SAML token, the trusting IdP is able to validate that SAML token because it is signed by a trusted IdP, and therefore authenticates the user. In this process, you can see that the user did not need to register with the trusting IdP, therefore saving a lot of registration procedures.

2.6.3 Third-Party Login

3rd party login supports customers of the IdM to enhance the reach of their websites by means of attracting users without forcing them to register new user accounts on their sites manually. 3rd party login allows users to register to the customers' sites with already existing digital identities from their favorite 3rd party identity providers, such as Google, Facebook or Yahoo. Thus, 3rd party login lowers the obstacles of registration processes and increases the number of successful business flows on the customers' sites.

2.6.4 Web Single Sign-On

As it is possible to configure several applications that shall be linked to his IdM, the main benefit for users is a single sign-on (SSO) to all these applications.

2.6.5 Hosted User Profile Management

The IdM offers hosted user profile storage with specific user profile attributes. Applications do not have to run and manage their own persistent user data storages, but instead can use the IdM user profile storage as a SaaS (Software as a Service) offering.

2.6.6 Multi-Tenancy

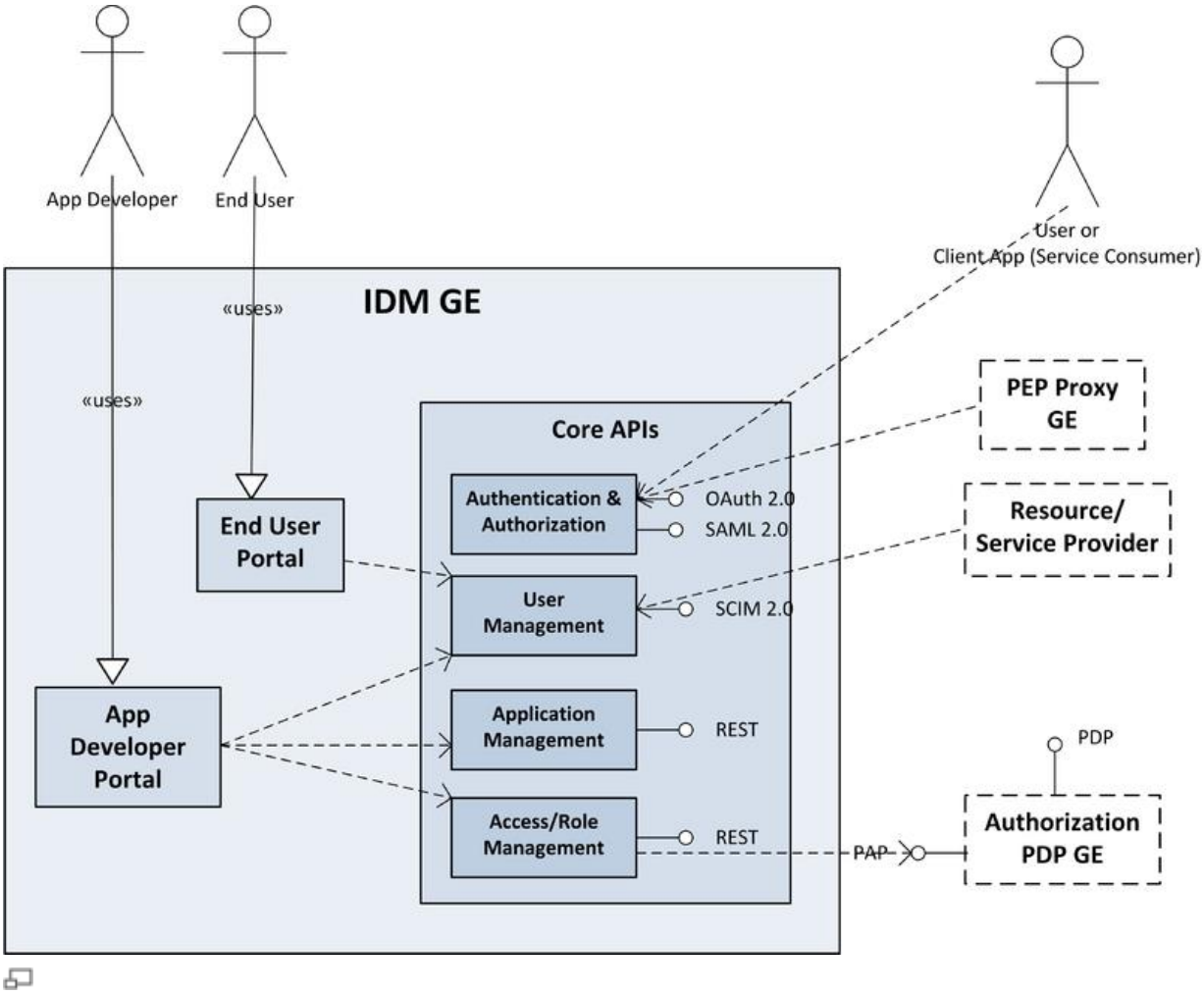
A multi-tenancy architecture refers to a principle in software architecture where a single software instance runs on a server, serving multiple client/customer organizations (tenants). Multi-tenancy contrasts with a multi-instance architecture where separate and dedicated software instances (or hardware systems) are set up for different client organizations. With a multi-tenant architecture, a software application is designed to virtually partition its data and configuration, and each client organization works with a customized virtual application instance. In a multi-tenancy environment, multiple customers share the same application, running on the same operating system, on the same virtualized hardware, with the same data storage mechanism. The distinction between the customers is achieved during application design, thus customers do not share or see each other's data. The concept allows each tenant to apply their own branding to login or registration UIs or for user self-services to create a user experience that is aligned with the one offered in a tenant application.

2.7 Main Interactions

We here define the mandatory and optional interfaces, including API and protocol standards, expected from IdM GE implementations; also who interacts with these interfaces, and possible interactions.

2.7.1 High-level IdM GE Architecture

This section provides an overview of the Identity Management Generic Enabler (IdM GE)'s interfaces. The general structure of a FIWARE IdM GE is sketched in the following figure:



Identity Management GE - High Level Architecture

The various modules and interactions shown on the figure are described in the next sections.

2.7.2 User interfaces

- **End User portal:** This is where end users self-register in the IdM with email address, password, etc. This is typically implemented as a Web user interface. End users may also review and modify their personal account data and maintain their privacy settings using this portal.

- **Application Developer Portal:** This is where developers can register and manage their applications, especially the client applications, including the application credentials. With such credentials, the application is able to authenticate to the IdM and participate in an authentication and authorization process (explained in more details in the Authentication and Authorization Interfaces section later on) to get access to a protected Service Provider. The developer can also manage access for his application, and in particular, define application-specific roles.

2.7.3 Management APIs

- **User management API:** Provides a REST API to create user accounts, retrieve and modify user attributes, delete user accounts. The interface must be compliant with SCIM 2.0[2] REST API. The user management API is typically used by web applications (or any kind of service provider), to retrieve extra information about their users.
- **Application management API:** REST API for managing applications (registering the application, retrieving and modifying application data such as credentials, deleting the application).
- **Access Management API:** REST API to manage roles globally or for a specific application. There are two aspects of role management involved here: defining the role permissions and assigning the roles to the users. The role permissions make up an authorization policy that may be pushed to the Authorization PDP GE via its PAP API. Please refer to *Basic Concepts* section of the Authorization PDP GE architecture description for more information.

2.7.4 Authentication and Authorization Interfaces

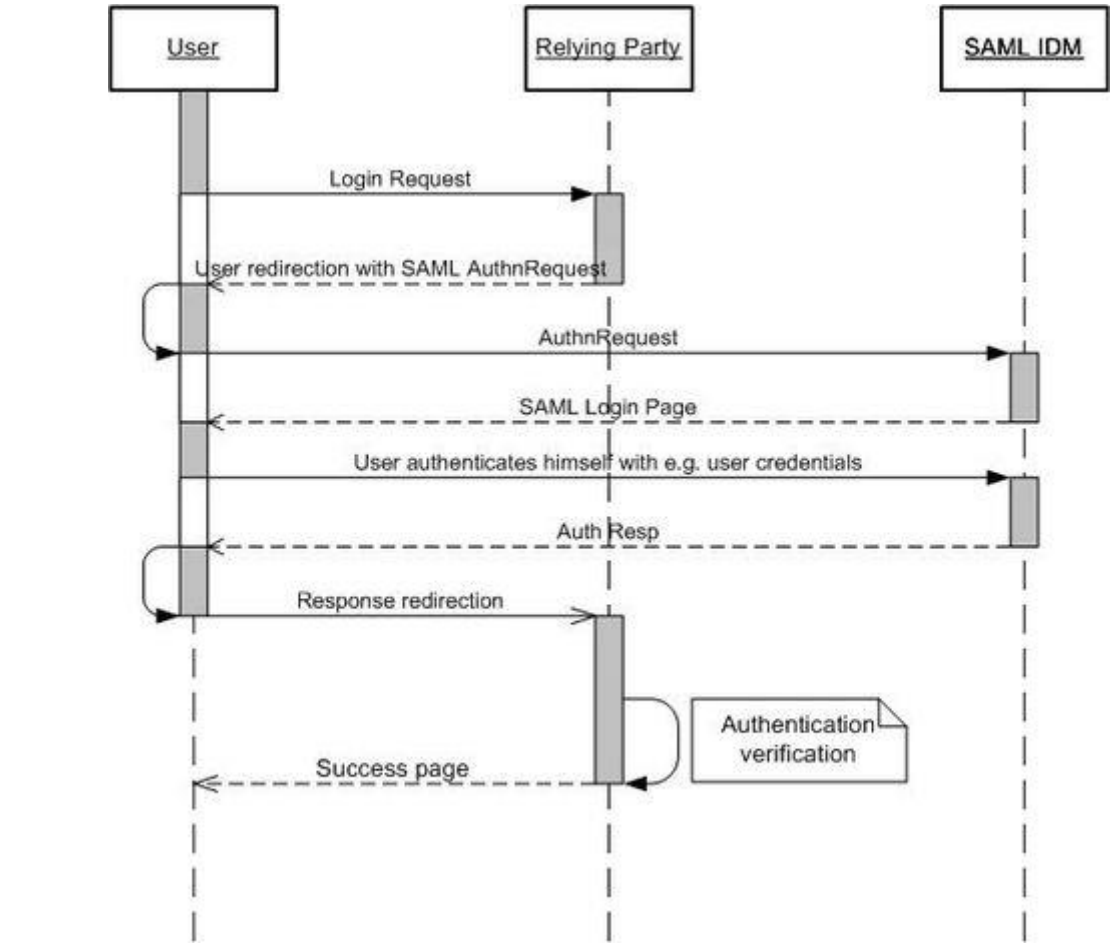
This section focuses on the standards used and made mandatory for the IdM GE's Authentication and Authorization interfaces, and SSO by extension, with the help of message flows and reference code examples, thus offering an easy implementation and usage of the Generic Enabler.

2.7.4.1 *SAML*

The Identity Management GE supports SAML 2.0[3] to provide federated identity, more specifically federated Single Sign-On and user attribute exchange between IdM systems (FIWARE IdM GE and other Identity Providers).
The advantages of SAML 2.0 are:

- Provides a means of exchanging data between security domains (i.e. the Identity Management GE and its federated service providers (relying parties))
- Provides the SSO feature for the federated service providers to the Users
- Service providers do not need to authenticate users themselves
- Provides security features such as digital signatures to certify the integrity of the exchanged data (and certified attributes)
- Standardized, non-proprietary protocol (e.g. also supported by Google)

The diagram below shows an example of a SAML interaction. Details of the request parameters and flows can be found in the SAML specification [3]. On the picture, the SAML IDM may be in fact the IdM GE in the context of FIWARE or another SAML Identity Provider that is federated with the FIWARE IdM GE.



Identity Management GE - SAML Authentication Flow

2.7.4.1.1 *Authentication Request*

In a SAML Web-SSO flow, whenever the user requests access to a service provider (Relying Party in the previous figure) for the first time or after his/her session cookie for this service has expired, in other words, with no valid session cookie, the service redirects the user - the user agent (typically the web navigator) - to a trusted Identity Provider (SAML IDM on the previous figure) with the following SAML authentication request. The main parameters are the time when the request is emitted (Issue Instant) and the service identifier as known by the IDM, so that the IDM knows, which relying party the user is requesting a SAML token for; and therefore, where to redirect the user after authentication, which cryptographic protection to apply on the SAML token, which SAML attributes to include in the token, and so on. Please see an example of such Request below:

```
<samlp:AuthnRequest
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="aaf23196-1773-2113-474a-fe114412ab72"
    Version="2.0"
    IssueInstant="2004-12-05T09:21:59Z"
```

```
        AssertionConsumerServiceIndex="0"
        AttributeConsumingServiceIndex="0">
    <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
    <samlp:NameIDPolicy
        AllowCreate="true"
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/></samlp:AuthnRequest>
```

2.7.4.1.2 *Authentication Response*

The user authenticates to the SAML IDM, and after successful authentication, the SAML IDM returns a SAML Authentication Response that contains the SAML assertion (aka SAML token) signed by the IDM. Besides the digital signature, it contains various statements, such as the time when it was issued, the IDM identifier, the validity period, the Relying Party for which this token may be used, the method of authentication used to obtain the token, etc. Please see an example below:

```
<samlp:Response
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="identifier_2"
    InResponseTo="identifier_1"
    Version="2.0"
    IssueInstant="2004-12-05T09:22:05Z"
    Destination="https://sp.example.com/SAML2/SSO/POST">
  <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode
        Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      ID="identifier_3"
      Version="2.0"
      IssueInstant="2004-12-05T09:22:05Z">
    <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
    <ds:Signature
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
    <saml:Subject>
      <saml:NameID
          Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">3f7b3dcf-1674-4ecd-
92c8-1544f346baf8</saml:NameID>
      <saml:SubjectConfirmation
          Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData
            InResponseTo="identifier_1"
            Recipient="https://sp.example.com/SAML2/SSO/POST"
            NotOnOrAfter="2004-12-05T09:27:05Z"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions
        NotBefore="2004-12-05T09:17:05Z"
        NotOnOrAfter="2004-12-05T09:27:05Z">
      <saml:AudienceRestriction>
        <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
      </saml:AudienceRestriction>
```

```
39.       </saml:Conditions>
40.       <saml:AuthnStatement
41.            AuthnInstant="2004-12-05T09:22:00Z"
42.            SessionIndex="identifier_3">
43.          <saml:AuthnContext>
44.
<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</saml:AuthnContextClassRef>
45.          </saml:AuthnContext>
46.       </saml:AuthnStatement>
47.    </saml:Assertion>
48. </samlp:Response>
```

The IDM redirects the user with this Response to the Relying Party, i.e. the service provider. The latter considers the user is properly authenticated because the SAML token received from the user is signed and issued by a trusted IDM; and the token proves that the user has successfully authenticated to this IDM.

2.7.4.2 *OAuth2*

The OAuth[4] standard is an authentication and authorization framework that initially addresses the scenario where you have to allow a website or application (*Consumer*) to access protected resources of an *End User* from a web service (*Service Provider*) via an API, without requiring this *End User* to disclose their Service Provider credentials to the Consumer. Then the standard was extended in its version 2.0 to address other more traditional use cases of web API authentication, in order to use the same framework for most common scenarios in this domain. We now dare to say that OAuth creates a generic methodology for web – especially RESTful – API authentication.

We use the term *Service Provider* here as an equivalent for the term *Resource Server* used in the RFC of OAuth 2.0[4] mentioned above, because we think it makes more sense to FIWARE developers with regards to what the IdM GE's OAuth capabilities can do for him/her. The term *Resource Server* may sound more restrictive in terms of possible use cases than OAuth actually addresses. For the record, OAuth 1.0 used the term *Service Provider* instead as well. The same comment goes for *Consumer*, as in *Service Consumer*, used here instead of the term *Client* in the OAuth 2.0 [4] specification.

OAuth 2.0 presents significant advantages compared to other delegation/authorization frameworks:

- A standardized protocol supported by a wider set of Service Providers (Facebook, Google, LinkedIn…)
- The end user grants access for a consumer to a specific resource by providing an access token to the consumer. The user may be offline, when the consumer accesses the resource, i.e. actually makes use of the access token.
- The end user is in full control of who can access his/her resources. The delegation of access can be limited in scope and time, and may be revoked any time by the user.
- Mitigation of security risks of sharing credentials with third party: Indeed, if not using OAuth, for the same kind of requirement, the third party application (Consumer) gives no choice to the user but to give away their credentials to the third party, with the associated risks listed below:
    - The third party (Consumer) is now storing the end user's credentials. In particular, it needs access to the user's password in clear text in order to

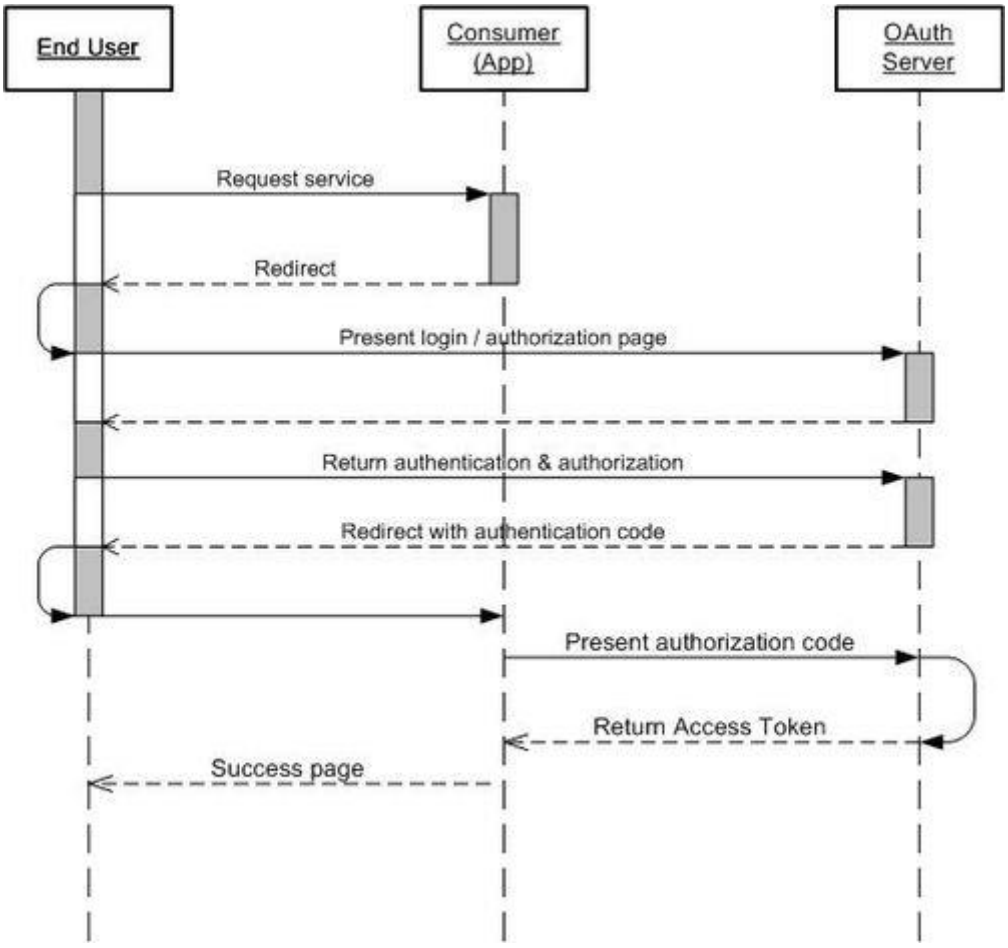
authenticate on his behalf to the Service Provider. This is different from the Service Provider's side, where only a hash of the password is needed for authenticating the user. As a result, if the third party is compromised, so are your credentials.
    - This solution works well for password authentication only. It is not compatible with other forms of authentication, more complex, such as the ones relying on a hardware device and/or biometrics.
    - The third party gets the same access as you, i.e. unlimited access to all your resources on the Service Provider.
    - Revocation of the third party's access by the user is painful, because it requires a change of password. But then it removes access from every other third party, which was not intended. Therefore, you have to go find and replace the password with the new one for all these other third parties that you still want to grant access to. In other words, the access revocation cannot be done for a specific third party and only this one.
- Different grant types for different scenarios:
    - **Authorization Code grant**: The most well-known when people think of OAuth, used whenever you log on some website with the Facebook/Twitter/Google login button. For example, if you click on the Google button on the website X (*Consumer*), you are redirected to Google (*OAuth server*) for sign-in, then asked if website X can access your Google name, email address, etc. (your *resources* at Google). If you say yes, you are redirected to website X with a special token called *authorization code*. Then the website X authenticates to Google (OAuth server) and then exchanges the *authorization code* for the actual *access token*, in the background, without the end-user being involved; and uses this *access token* to get user info from Google API (*Service Provider)*. This process is explained in further details in the next section. You cannot fail to notice that in this example, the OAuth server and Service Provider are both Google's. However, in FIWARE, the Service Provider may be completely distinct from the IdM GE that plays the role of the OAuth Server, i.e. not owned by the same organization at all.
    - **Implicit grant**: This is similar to the *Authorization code grant*, except the access token is returned to the Consumer right away, when the user is redirected back, instead of the authorization code. Therefore, the step where the Consumer authenticates to the OAuth server is skipped. As a result, the authorization flow is simpler and the Consumer does not need to store its credentials for authenticating itself to the OAuth server at all. This is particularly useful for lightweight applications such as JavaScript embedded in the browser that are not capable of storing/persisting secret credentials.
    - **Resource owner credentials grant**: this is the old traditional way described previously in the *Mitigation of security risks of sharing credentials with third party*, where the Consumer asks the username and password of the end user (*Resource Owner* in OAuth2 terms). This is only suitable for trusted clients such as the user's own mobile phone or PC. It can be used for other specific use cases such as legacy applications for which implementing the authorization code grant requires too much effort, but then you should be

perfectly aware of the security implications mentioned earlier and in the OAuth specification.

- **Client credentials grant:** similar to the *Resource owner credentials grant*, except the Consumer (i.e. *Client* in OAuth 2 terms) does not ask for the user's credentials, but uses its own. In other words, this addresses the traditional client-server authentication scenario where the client authenticates to the server (OAuth server in this case) on its own (with its own credentials). In the context of OAuth, the client gets an *access token* for the particular *Service Provider* specified in the authentication request, as a result.

Implementations of this GE must support at least the authorization code grant (§4.1 of OAuth 2.0[4] specification) and resource owner password credentials grant (§4.3 of OAuth 2.0[4] specification). Optionally implicit grant (§4.2 of OAuth 2.0[4] specification) may also be supported. IDM GEis should also allow configuring trusted FIWARE components to access protected resources using (§4.4 of OAuth 2.0[4] specification) client credentials grant.

Below it is shown an example of OAuth2 interaction for the *Authorization Code grant*, already summarized in the previous paragraph. Details of the request parameters and flows can be found in the protocol specification[4]. In the context of FIWARE, the OAuth Server role shown on the picture is played by the IdM GE.



Identity Management GE - OAuth 2.0 Authentication Flow

2.7.4.2.1 *Get Request Token*

```
https://api.login.<xyz.com>/oauth/v2/get_request_token?oauth_nonce=ce2130523f788f313f76314ed3965ea6
&oauth_timestamp=1202956957
&oauth_consumer_key=12345689101112131415161718192O
&oauth_signature_method=plaintext
&oauth_signature=abcdef
&oauth_version=1.0
&xoauth_lang_pref="en-us"
&oauth_callback="http://yoursite.com/callback"
```

2.7.4.2.2 *Get Request Token Response*

```
oauth_token=z4ezdgj
&oauth_token_secret=47ba47e0048b7f2105db67df18ffd24bd072688a
&oauth_expires_in=3600
&xoauth_request_auth_url=https%3A%2F%2Fapi.login.<xyz.com>%2Foauth%2Fv2%2Frequest_auth%3Foauth_token%3Dz4ezdgj
&oauth_callback_confirmed=true
```

2.7.4.2.3 *Get Access Token Request*

```
https://api.login.<xyz.com>/oauth/v2/request_auth?oauth_token=j5nyp6
```

2.7.4.2.4 *Get Access Token Response*

```
http://yoursite.com/callback?oauth_verifierer=svmhhd
```

2.7.4.2.5 *Exchange Token Request*

```
https://api.login.<xyz.com>/oauth/v2/get_token?oauth_consumer_key=dj0yJmk9NG5USlVvTlZsZEpnJmQ9WVdrOVQwa
zFPRUozTkc4bWNHozlNVE13TXprM01UUTBNZy0tJnM9Y29uc3VtZXJzZWNyZXQmeD1kNg--
```

&oauth_signature_method=PLAINTEXT

&oauth_version=1.0

&oauth_verifier=svmhhd

&oauth_token=gugucz&oauth_timestamp=1228169662

&oauth_nonce=8B9SpF

&oauth_signature=5f78507cf0acc38890cf5aa697210822e90c8b1c%261fa61b464613d0d32de80089fe099caf34c9dac5

2.7.4.2.6 *Exchange Token Response*

oauth_token=A%3DqVDHXBngo1tEtzox.JMhzd91Rk99.39Al7hos3J80mm1j_3nGP4BiilL777vUj2rsPLj1cZw.srbisvw.cz42Lzmlxt

H0Kk9mkXilvS1ll5lNoMKXO5zy5YG4vO3fbGKewp7IESYMIdEi4Md7SroYiv6kBCEjqB4jXr0.8XsMvOlQgZ.aKNKXwc2sv3n4BOZxs

54tzXV6rGNpEHZUaj9CovPUo44isTgs9FnLIKpXFCU4Jq1BB3_IOTFBNf1vtf5vSxaxe_L5dUhr.i15Hx0LTZ2tlsWeDcActSGGBWVc

vytPF3cK9mDWy44baBgCVI3AEbGCqg.NGhDPqOh1ZHfKFtYlBZfG4xf2n..CdxcM5x4INxnVz2.biMkfhfkw8haJuR0RaUY37lBxZ9z

2.7.4.3 ***Username/Password***

Besides the authentication/authorization mechanisms described above, the basic username/password authentication method is supported. The username and password are verified against the IdM's backend user database.

2.8 Identity Management GE infrastructure

Beyond the modules and main interactions with other GEs and components described in the previous sections, the IdM GE may depend on additional security components to secure access to the GE interfaces. For example, the IdM GE requires a Public Key Infrastructure (PKI) to issue and manage SSL certificates for transport-level security (authentication, confidentiality and integrity) of communications with the IdM (e.g. over HTTPS), and message-level security such as digital signatures of SAML assertions issued by the IdM.

The IdM GE may also depend on non-security components such as a mail server to communicate with the users (e.g. for email address confirmation, password recovery...).

2.9 References

1. ↑ F. Schneider, Trust in Cyberspace, 1998. *Trust in Cyberspace* . National Academy Press, Washington, DC, USA. 1998.
2. ↑ System for Cross-Domain Identity Management - http://tools.ietf.org/html/draft-ietf-scim-core-schema-02 and http://tools.ietf.org/html/draft-ietf-scim-api-02
3. ↑ 3.0 3.1 Security Assertion Markup Language - http://saml.xml.org/saml-specifications, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20
4. ↑ 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 IETF RFC 6749 - https://tools.ietf.org/html/rfc6749, http://oauth.net

2.10 Detailed Specifications

The detailed API specification is available on the APIary website: http://docs.keyrock.apiary.io/

2.11 Re-utilised Technologies/Specifications

The Identity Management GEis are based on the following technologies:

- SAML 2.0 protocol,
- RFC 6749 - The OAuth 2.0 Authorization Framework,
- SCIM protocol.

2.12 Terms and definitions

This section comprises a summary of terms and definitions introduced during the previous sections. It intends to establish a vocabulary that will be help to carry out discussions internally and with third parties (e.g., Use Case projects in the EU FP7 Future Internet PPP). For a summary of terms and definitions managed at overall FIWARE level, please refer to FIWARE Global Terms and Definitions

- **Access control**: is *the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner* (ITU-T Recommendation X.800). More precisely, access control is the protection of resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entities according to that policy (RFC 2828).

- **Account**: A (user) account is “typically a formal business agreement for providing regular dealings and services between principal sand business service providers.” OASIS Security Assertion Markup Language (SAML).

- **Authentication (AuthN)**: We adopted the following definition of authentication from

RFC 3588: "Authentication is “the act of verifying the identity of an entity (subject)”.

TrustInCyberspace adds the term “level of confidence” to this definition:
Authentication is the process of confirming a system entity’s asserted identity with a specified, or understood, level of confidence.” This definition holds all necessary parts to examine authentication in broad sense. First of all it does not narrow the authentication to human users, but refers to a generic “system entity”. See authentication reference architecture description for a closer look at different identities that could be authenticated.
Secondly it introduces the often neglected concept of “level of confidence” which applies to each authentication of an identity. No computer program or computer user can definitely prove the identity of another party. There is no authentication method that can be secured against any possible identity-theft attack, be it physical or non-physical. It is only possible to apply one or more tests, which, if passed, have been previously declared to be sufficient to go on further. The problem is to determine which tests are sufficient, and many such are inadequate.
The original Greek word originates from the word 'authentes'='author'. This leads to the general field of claims and trust management, because authentication could also mean to verify the “author” / issuer of any claim.
The confirmation or validation process of authentication is actually done by presenting some kind of proof. This proof is normally derived from some kind of secret hold by the principal. In its simplest form the participant and the authentication authority share the same secret. More advanced concepts rely on challenge/response mechanisms, preventing the secrets to be transmitted. Refer to Authentication Technologies for a detailed list of authentication methods used today.
As stated above, each authentication method assures only some level of trust in the claimed identity, but none could be definite. Therefore it makes sense to distinguish the different authentication methods by an associated assurance level, stating the level of trust in the authentication process.
As this assurance level depends not only on the technical authentication method, but also on the overall computer system and even on the business processes within the organization (provisioning of identities and credentials), there is no ranking of the authentication methods here.

- **Authentication protocol**: "Over-the-wire authentication protocols are used to exchange authentication data between the client and server application. Each authentication protocol supports one or more authentication methods. The OATH reference architecture provides for the use of existing protocols, and envisions the use of extended protocols which support new authentication methods as they are defined." (OATH)

- **Federation**: The term federation “is used in two senses - "The act of establishing a relationship between two entities. An association comprising any number of service providers and identity providers.” OASIS Security Assertion Markup Language (SAML).

  “A federation is a collection of realms that have established a producer-consumer

relationship whereby one realm can provide authorized access to a resource it manages based on an identity, and possibly associated attributes, that are asserted in another realm.
Federation requires trust such that a relying party can make a well-informed access control decision based on the credibility of identity and attribute data that is vouched for by another realm." WS-Federation @ IBM
Remark: Federation according to WS-Federation @ IBM is similar to the concept of a Circle of Trust.

- **Identifier:** Identifiers can be understood as a dedicated, publicly known attribute of an identity that refers to that identity only. Typically, identifiers are valid within a specific domain. Special types of identifiers are valid globally, due to the use of popular domain naming and resolution protocols such as DNS, which implies addressing capabilities to the identity. OASIS Security Assertion Markup Language (SAML) defines identifier as follows:

  An identifier is "a data object (for example, a string) mapped to a system entity that uniquely refers to the system entity. A system entity may have multiple distinct identifiers referring to it. An identifier is essentially a "distinguished attribute" of an entity."

- **Identity** (Digital): The term identity and its meaning have been discussed controversially in the "identity community" for many years. Until now, there is no commonly agreed definition of that notion. The IdM and AAA reference architecture applies the following three definitions of identity.

  The Identity Gang defines the term digital identity as follows:
  A digital identity is "a digital representation of a set of Claims made by one party about itself or another digital subject."
  The following comments were added:
  A digital identity is just one set of claims about a digital subject. For any given digital subject there will typically be many digital identities.
  A digital identity can be created on the fly when a particular identity transaction is desired or persistent in a data store to provide a representation that can be referenced.
  A digital identity may contain claims made by multiple claimants.
  A digital identity may be signed by a digital identity provider to provide assurance to a relying party.
  This definition emphasizes two facts:
  Normally, a principal (subject) has multiple digital identities or personas.
  Identities are made out of attributes (claims).
  Therefore, the scope of identity management in the reference architecture has two viewpoints: For once it focuses on identities and personas itself, and on the other side, it deals with the attributes of these identities and personas.
  The Liberty Alliance Project (LAP) defines digital identity as follows:
  Digital identity is "the essence of an entity. One's identity is often described by one's

characteristics, among which may be any number of identifiers. A Principal may wield one or more identities."
RSA uses the following definition of digital identity:
"Digital identity consists of an identity assertion and the characteristics, sometimes called attributes that are collected or observed through our computerized relationships. It is often as simple as a user name and password."
The definition of RSA adds one important aspect to the identity discussion: Even the simplest user name and password combinations without any additional attributes or claims constitute an identity.

- **Identity context**: is "the surrounding environment and circumstances that determine meaning of digital identities and the policies and protocols that govern their interactions." (Identity Gang)

- **Identity management (IdM)**: comprises "the management of identity information both internally and when it is passed from one entity to another." Open Mobile Alliance (OMA)

- **Identity provider**: The Open Mobile Alliance (OMA) defines the term identity provider (IdP) as follows - An identity provider is "a special type of service provider […] that creates, maintains, and manages identity information for principals, and can provide […] assertions to other service providers within an authentication domain (or even a circle of trust)."

  Another notion defines identity provider as "an agent that issues a digital identity [that] is acting on behalf of an issuing Party." (Identity Gang)
  The following definition of identity provider descends from WS-Federation @ IBM: "An identity provider is an entity that acts as an authentication service to end requestors and as data origin authentication service to service providers […]. Identity providers are trusted (logical) 3rd parties which need to be trusted both by the requestor […] and the service provider which may grant access to valuable resources and information based upon the integrity of the identity information provided by the identity provider."
  The Identity Provider is part of the Identity Management infrastructure.

- **Single sign-on**: is "From a Principal's perspective, single sign-on encompasses the capability to authenticate with some system entity—[…] an Identity Provider - and have that authentication honored by other system entities, [termed] Service Providers […]. Note that upon authenticating with an Identity Provider, the Identity Provider typically establishes and maintains some notion of local session state between itself and the Principal's user agent. Service Providers may also maintain their own distinct local session state with a Principal's user agent." Liberty Alliance Project (LAP)

3 FIWARE.OpenSpecification.Security.PEPProxy

Name	FIWARE.OpenSpecification.Security.PEP Proxy Generic Enabler
Chapter	Security,
Catalogue-Link to Implementation	Wilma
Owner	UPM , Alvaro Alonso

3.1 Preface

Within this document you find a self-contained open specification of a FIWARE generic enabler, please consult as well the FIWARE Product Vision, the website on http://www.fiware.org and similar pages in order to understand the complete context of the FIWARE platform.

3.2 Copyright

3.3 Legal Notice

Please check the following Legal Notice to understand the rights to use these specifications.

3.4 Overview

A Policy Enforcement Point (PEP) is a component that protects resources against *unauthorized* access; *unauthorized* meaning: *which does not comply with the access control policy applicable for these resources*. It represents the final piece of the larger suite of Identity and Access Management GEs, of which we already mentioned the IdM and the Authorization PDP in previous sections. The PEP is the one intercepting each access request to the resource, but relies on the IdM to authenticate the request, and on the PDP to authorize it (deny of permit). In particular, on the contrary to the IdM and PDP, the PEP understands the particular API requests and protocols for accessing the resource under its protection. It also knows how to query the authentication and authorization APIs of the IdM and PDP. Last but not least, the PEP knows where and how to extract which information from the request that is necessary for authentication and authorization, and how to send it to the IdM and PDP to do just that.

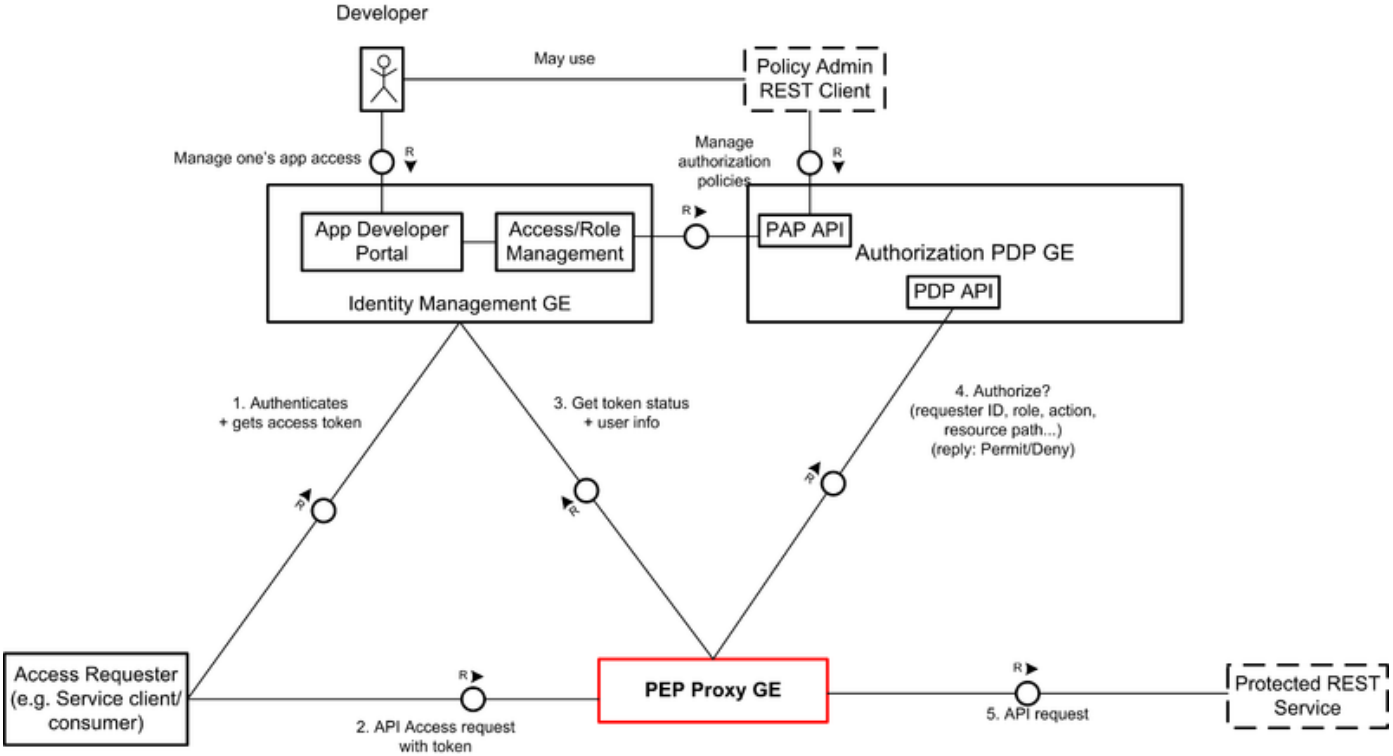
In the case of the PEP Proxy GE, the protected resource is a HTTP service (API), e.g. a REST service, for which the GE plays the role of reverse proxy and is deployed as such. In this configuration, it is critical that any access request goes through the PEP proxy before reaching the protected service. To prevent any bypass, the application developer, usually with the help of the system and network administrators of the IT infrastructure where the application is deployed, makes sure that the proper

perimeter security controls, e.g. firewall rules, are in place, and the application server (where the protected application is deployed) is not listening on ports reachable from outside except the one used by the PEP proxy. If the PEP proxy is deployed locally on the same host, then the application server only needs to listen locally. If the PEP Proxy is deployed on a different network from the application, it may result in communications going from the PEP Proxy server to the application server over an untrusted or uncontrolled network, e.g. Internet. In this case, it is critical to set up a secure channel between the two servers or two networks to protect the confidentiality and integrity of the communications, e.g. with TLS. We recommend that security-unaware developers avoid this last situation whenever possible.

Besides the PEP Proxy's own TLS server certificate setup, the developer must configure the PEP Proxy with the HTTPS URL to the IdM GE for token validation. As the communications with the IdM GE must be secured, the PEP proxy must be configured to trust the IdM GE's TLS certificate or the issuer certificate, i.e. a Certificate Authority (CA). Likewise, the PEP Proxy configuration must include the HTTPS URL to the Authorization PDP GE for authorization and trust the PDP GE's certificate or issuing CA. Besides, the developer must configure the URL to the protected service API endpoint, so that the PEP Proxy knows where to forward the requests.

Furthermore, we assume that the developer has registered his application in the IdM GE, including the credentials for authenticating the application, and he/she (or any application security administrator in the developer's organization) has configured an authorization policy for this application (REST service), either via the graphical interface of the Developer Portal, calling the Authorization PDP GE's PAP API as backend; or sending (with a REST client) the policy directly in XACML format to the same PAP API. Please refer to the Authorization PDP GE (Overview) section for more information.

The PEP proxy GE is now able to intercept each incoming access request to the service and perform the authentication and authorization workflow according to the figure below:



PEP Proxy - Architecture overview

1. It is expected that the requester has authenticated to the IdM GE using OAuth2 flow, and got an access token from the IdM GE as a result. (For more information, please refer to the Identity Management GE architecture description.)
2. The requester sends the API request with the token included as the PEP Proxy GE expects, that is to say in a specific HTTP header in a specific format – depending on the GE implementation and configuration. For example, the PEP Proxy GE reference implementation supports access tokens in the Authorization header as defined by the OAuth 2.0 Bearer Token standard (IETF RFC 6750). In any case, In order to protect the confidentiality of access tokens in transit, the access request must be sent by the requester to the PEP over TLS (HTTPS), following the MUST requirement in Section 10.3 of OAuth 2.0 specification (IETF RFC 6749). This implies that the PEP proxy must have TLS enabled with a valid server certificate.
3. When the PEP Proxy receives an access request, it extracts the access token from the specific request header mentioned previously and sends it to the IdM GE for validation. If it is valid, the IdM GE returns the validation result and other token-related information, such as information about the authenticated user (e.g. user role). If the token turns out not valid, the request is not authenticated therefore denied.
4. The PEP Proxy sends to the Authorization PDP API an XACML authorization decision request that contains this IdM-issued information with other information about the access request: Requested resource ID, action ID (HTTP method), etc. For instance, for a REST API request, the PEP Proxy sends the URL requested for the resource, the HTTP method for the action ID, and authenticated user attributes. The Authorization PDP GE computes the authorization decision – Permit or Deny – and returns it to the PEP
5. If PDP's decision is Permit, the PEP forwards the API request to the protected service, and forwards the response back to the requester. If the decision was Deny, the PEP denies the request, for instance replies with a HTTP response 403 (Forbidden).

3.5 Basic Concepts

The main concepts of this section are:

- **Policy Enforcement Point** (PEP) that this PEP Proxy GE represents;
- **Policy Decision Point** (PDP), that this PEP Proxy enforces decisions of.

Please refer to the *Basic Concepts* sub-section of the Authorization PDP GE architecture description for more information.

3.6 Main Interactions

The figure in the Overview section gave an overview of the GE's main interactions. To summarize on the subject, we can say that the PEP Proxy GE interacts with two components in order to check authentication and authorization:

- **Identity Management GE**: When PEP Proxy receives a request, it retrieves the authentication token from the specific header and validates it with the Identity Management GE.
- **Authorization PDP GE**: If the component is configured to check not only the authentication but also the authorization, PEP Proxy will check with Authorization PDP if the user (from the token) has the correct permissions to access the resource specified in the request.

3.7 Basic Design Principles

- Compatibility with REST APIs;
- HTTP Reverse-proxy capabilities;
- Gradual authentication and authorization with following options:
  - Simple reverse-proxy functionality for whitelist of URLs (no authentication/authorization);
  - Authentication only: the step involving the PDP GE may be skipped for some use case;
  - Full authentication and authorization: includes the step involving the PDP GE.
- Transport-Layer Security support:
  - As a server: for securing communications between clients and the PEP Proxy;
  - As a client: for securing communications between the PEP Proxy and the IdM GE and PDP GE.
- OAuth Bearer token (RFC 6750) support.

3.8 Detailed Specifications

3.8.1 Open API Specification

- PEPProxy.Open API Specification

3.8.2 References

- FIWARE.OpenSpecification.Security.IdentityManagement
- FIWARE.OpenSpecification.Security.AuthorizationPDP
- [OAuth2] OAuth 2.0 Authorization Framework
- [XACML3] XACML 3.0

3.9 Re-utilised Technologies/Specifications

The PEP Proxy GE is based on RESTful Design Principles. The technologies and specifications used in this GE are:

- RESTful web services
- HTTP/1.1
- OAuth 2.0
- JSON and XML data serialization formats.

3.10 Terms and definitions

This section comprises a summary of terms and definitions introduced during the previous sections. It intends to establish a vocabulary that will be help to carry out discussions internally and with third parties (e.g., Use Case projects in the EU FP7 Future Internet PPP). For a summary of terms and definitions managed at overall FIWARE level, please refer to FIWARE Global Terms and Definitions

- **Access control**: is *the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner* (ITU-T Recommendation X.800). More precisely, access control is the protection of resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entities according to that policy (RFC 2828).

- **Account**: A (user) account is “typically a formal business agreement for providing regular dealings and services between principal sand business service providers.” OASIS Security Assertion Markup Language (SAML).

- **Authentication (AuthN)**: We adopted the following definition of authentication from RFC 3588: "Authentication is “the act of verifying the identity of an entity (subject)”

  TrustInCyberspace adds the term “level of confidence” to this definition:
  Authentication is the process of confirming a system entity’s asserted identity with a specified, or understood, level of confidence.” This definition holds all necessary parts to examine authentication in broad sense. First of all it does not narrow the authentication to human users, but refers to a generic “system entity”. See authentication reference architecture description for a closer look at different identities that could be authenticated.
  Secondly it introduces the often neglected concept of “level of confidence” which applies to each authentication of an identity. No computer program or computer user can definitely prove the identity of another party. There is no authentication method that can be secured against any possible identity-theft attack, be it physical or non-physical. It is only possible to apply one or more tests, which, if passed, have been previously declared to be sufficient to go on further. The problem is to determine which tests are sufficient, and many such are inadequate.
  The original Greek word originates from the word 'authentes'='author'. This leads to the general field of claims and trust management, because authentication could also

mean to verify the “author” / issuer of any claim.
The confirmation or validation process of authentication is actually done by presenting some kind of proof. This proof is normally derived from some kind of secret hold by the principal. In its simplest form the participant and the authentication authority share the same secret. More advanced concepts rely on challenge/response mechanisms, preventing the secrets to be transmitted. Refer to Authentication Technologies for a detailed list of authentication methods used today.
As stated above, each authentication method assures only some level of trust in the claimed identity, but none could be definite. Therefore it makes sense to distinguish the different authentication methods by an associated assurance level, stating the level of trust in the authentication process.
As this assurance level depends not only on the technical authentication method, but also on the overall computer system and even on the business processes within the organization (provisioning of identities and credentials), there is no ranking of the authentication methods here.

- **Authentication protocol**: "Over-the-wire authentication protocols are used to exchange authentication data between the client and server application. Each authentication protocol supports one or more authentication methods. The OATH reference architecture provides for the use of existing protocols, and envisions the use of extended protocols which support new authentication methods as they are defined." (OATH)

- **Federation**: The term federation “is used in two senses - "The act of establishing a relationship between two entities. An association comprising any number of service providers and identity providers.” OASIS Security Assertion Markup Language (SAML)

  “A federation is a collection of realms that have established a producer-consumer relationship whereby one realm can provide authorized access to a resource it manages based on an identity, and possibly associated attributes, that are asserted in another realm.
  Federation requires trust such that a relying party can make a well-informed access control decision based on the credibility of identity and attribute data that is vouched for by another realm.” WS-Federation @ IBM
  Remark: Federation according to WS-Federation @ IBM is similar to the concept of a Circle of Trust.

- **Identifier**: Identifiers can be understood as a dedicated, publicly known attribute of an identity that refers to that identity only. Typically, identifiers are valid within a specific domain. Special types of identifiers are valid globally, due to the use of popular domain naming and resolution protocols such as DNS, which implies addressing capabilities to the identity. OASIS Security Assertion Markup Language (SAML) defines identifier as follows:

  An identifier is “a data object (for example, a string) mapped to a system entity that uniquely refers to the system entity. A system entity may have multiple distinct

identifiers referring to it. An identifier is essentially a "distinguished attribute" of an entity."

- **Identity** (Digital): The term identity and its meaning have been discussed controversially in the "identity community" for many years. Until now, there is no commonly agreed definition of that notion. The IdM and AAA reference architecture applies the following three definitions of identity.

  The Identity Gang defines the term digital identity as follows:
  A digital identity is "a digital representation of a set of Claims made by one party about itself or another digital subject."
  The following comments were added:
  A digital identity is just one set of claims about a digital subject. For any given digital subject there will typically be many digital identities.
  A digital identity can be created on the fly when a particular identity transaction is desired or persistent in a data store to provide a representation that can be referenced.
  A digital identity may contain claims made by multiple claimants.
  A digital identity may be signed by a digital identity provider to provide assurance to a relying party.
  This definition emphasizes two facts:
  Normally, a principal (subject) has multiple digital identities or personas.
  Identities are made out of attributes (claims).
  Therefore, the scope of identity management in the reference architecture has two viewpoints: For once it focuses on identities and personas itself, and on the other side, it deals with the attributes of these identities and personas.
  The Liberty Alliance Project (LAP) defines digital identity as follows:
  Digital identity is "the essence of an entity. One's identity is often described by one's characteristics, among which may be any number of identifiers. A Principal may wield one or more identities."
  RSA uses the following definition of digital identity:
  "Digital identity consists of an identity assertion and the characteristics, sometimes called attributes that are collected or observed through our computerized relationships. It is often as simple as a user name and password."
  The definition of RSA adds one important aspect to the identity discussion: Even the simplest user name and password combinations without any additional attributes or claims constitute an identity.

- **Identity context**: is "the surrounding environment and circumstances that determine meaning of digital identities and the policies and protocols that govern their interactions." (Identity Gang)

- **Identity management (IdM)**: comprises "the management of identity information both internally and when it is passed from one entity to another." Open Mobile Alliance (OMA)

- **Identity provider**: The Open Mobile Alliance (OMA) defines the term identity provider (IdP) as follows - An identity provider is "a special type of service provider […] that creates, maintains, and manages identity information for principals, and can provide […] assertions to other service providers within an authentication domain (or even a circle of trust)."

  Another notion defines identity provider as "an agent that issues a digital identity [that] is acting on behalf of an issuing Party." (Identity Gang)
  The following definition of identity provider descends from WS-Federation @ IBM: "An identity provider is an entity that acts as an authentication service to end requestors and as data origin authentication service to service providers […]. Identity providers are trusted (logical) 3rd parties which need to be trusted both by the requestor […] and the service provider which may grant access to valuable resources and information based upon the integrity of the identity information provided by the identity provider."
  The Identity Provider is part of the Identity Management infrastructure.

  Single sign-on: is "From a Principal's perspective, single sign-on encompasses the capability to authenticate with some system entity—[…] an Identity Provider - and have that authentication honored by other system entities, [termed] Service Providers […]. Note that upon authenticating with an Identity Provider, the Identity Provider typically establishes and maintains some notion of local session state between itself and the Principal's user agent. Service Providers may also maintain their own distinct local session state with a Principal's user agent." Liberty Alliance Project (LAP)

4 FIWARE.OpenSpecification.Security.AuthorizationPDP

Name	FIWARE.OpenSpecification.Security.AuthorizationPDP
Chapter	Security,
Catalogue-Link to Implementation	AuthZForce
Owner	THALES , Cyril Dangerville (TS)

4.1 Preface

Within this document you find a self-contained open specification of a FIWARE generic enabler, please consult as well the FIWARE Product Vision, the website on http://www.fiware.org and similar pages in order to understand the complete context of the FIWARE platform.

4.2 Copyright

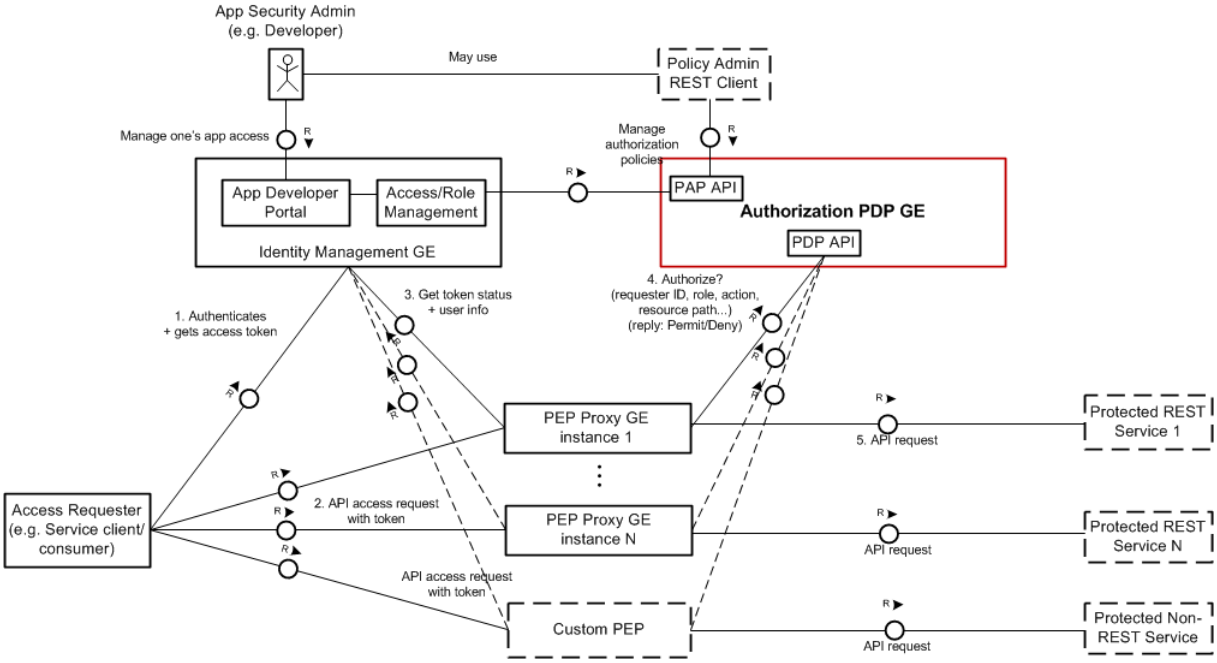
4.3 Legal Notice

4.4 Overview

The Authorization PDP Generic Enabler provides two main features:

- **Authorization policy decision evaluation**: This is the main feature of this GE as a PDP. Indeed, PDP stands for Policy Decision Point and its main feature consists to evaluate authorization decisions based on XACML policies and attributes related to a given access request (e.g. requester's identity, requested resource, requested action), following the policy evaluation logic defined in the XACML standard. This feature is provided to external clients through a REST API that we call the PDP API, where PDP is short for the term *Policy Decision Point* defined by the XACML standard.
- **Authorization policy administration**: creation, retrieval, update and removal of XACML policies. This feature is provided to external clients through a REST API that we call the PAP API, where PAP is short for the term *Policy Administration Point* defined by the XACML standard. This feature is necessary to support the previous feature. Indeed, it allows policy administrators (such as application developers potentially) to configure the XACML policies to be evaluated by the GE when calling the PDP API.

For more information on XACML, refer to the *Basic Concepts* section.
To have an overview of how the Authorization PDP GE can be used in FIWARE, especially by developers, and how it fits in the FIWARE architecture, please consider the figure below.



Authorization PDP - Architecture overview

To understand the various interactions going on in this picture, we will go through a typical scenario. Let us follow a typical scenario where some developer wants to define an authorization policy for his/her service, e.g. a RESTful web service, and then enforce this policy to control access to his/her application. We assume that the developer has registered his/her service like any other application in the Identity Management GE via the Application Developer Portal (see the Identity Management GE Architecture description for more information).

1. Via the Application Developer Portal, the developer can define an authorization policy for his/her service using a GUI (graphical user interface). In this case, we can say that the developer plays the role of *Application Security Administrator* pictured on the above diagram. This role may be played by another person in the developer's organization as well, such as a security officer or security architect working with the developer on the application. In any case, with the Developer Portal, this App Security Admin may define user roles specific to his/her service with the associated permissions expressed for example in a RESTful way (a permission HTTP method, and a resource URL path, '*' used as a wildcard):
    1. Role 'Reader' can only do GET actions on resource /docs/*
    2. Role 'Publisher' can only do GET, POST and PUT on resource /docs/*
    3. Role 'Administrator' can do any operation on any resource (/*).
2. When the App Security Admin (e.g. developer) has defined the access policy and decides to save and apply it, the IdM GE's Access management module is called by the Developer Portal to process this policy. The module converts into a XACML policy and sends it to the PAP API of the Authorization PDP GE.

3. The developer (or App Security Admin in general) may assign each of the defined roles to users of his/her choice.
4. The developer deploys an instance of the PEP Proxy GE in front of his/her service (pictured as *Protected Rest Service* on the diagram above), as a HTTP reverse-proxy, and configures it with the URL to the IdM GE, the URL to the Authorization PDP GE (more specifically the PDP API endpoint), and the URL to the service API endpoint of his service. Note that in order to protect the communications with the PEP, especially on a production system, it is necessary to enable SSL on the PEP and therefore configure the SSL certificate and other SSL settings. For more information on this part, please refer to the PEP Proxy GE architecture description.

The developer publishes the endpoint of the PEP Proxy instance as the new endpoint of his/her service, and makes sure it cannot be bypassed, usually by means of network filtering (e.g. firewall).
Everything is now in place to enforce access control. From this point forward, every access request to the service (REST service in this case) goes like this:

1. We assume that the access requester got a valid access token from the IdM GE before sending any request. It gets this token from the IdM GE (corresponding to the URL defined in the PEP configuration earlier). For more information on the process by which the requester may obtain an access token from the IdM, please refer to the Identity Management GE Architecture description.
2. The access requester sends a request to the new published endpoint of protected service, which happens to be the PEP proxy, with the access token included in a specific HTTP header. For more details on this step, please refer to the PEP Proxy GE architecture description.
3. The request is intercepted by the PEP Proxy. If the request does not have an access token (in the HTTP header) as expected by the PEP Proxy, the request is considered not authenticated and rejected by the PEP. Otherwise, the PEP extracts the access token from the header and sends it in a request to the IdM to validate the token, using the IdM token API. If the token is valid, the IdM replies with extra authorization info about the token, including the authenticated requester's attributes known by the IdM, e.g. ID and role. For more details on this step, please refer to the PEP Proxy GE architecture description.
4. The PEP sends a XACML Request to the Authorization PDP API with all this information about the requester and the access request, such as the requested action (HTTP method) and resource (URL path). The Authorization PDP GE evaluates the policy (defined by the developer and pushed via the Authorization PDP GE's PAP API earlier) for the input XACML Request received from the PEP. At the end of the evaluation, the PDP GE finally returns an authorization decision (Permit or Deny) to the PEP.
5. Depending on this decision, the PEP blocks (if Deny decision) or forwards (if Permit decision) the request to the service (REST service in this case). The application replies and the PEP forwards back the response to the requester, at last.

If you want to go into further details on the main interactions between the Authorization PDP GE and the other components shown on the picture above, please refer to the dedicated section *Main Interactions*.

4.5 Basic Concepts

It is useful at this point to give a quick overview of the OASIS XACML standard as it defines the essential parts of the GE features and APIs.
We can summarize it in three parts:

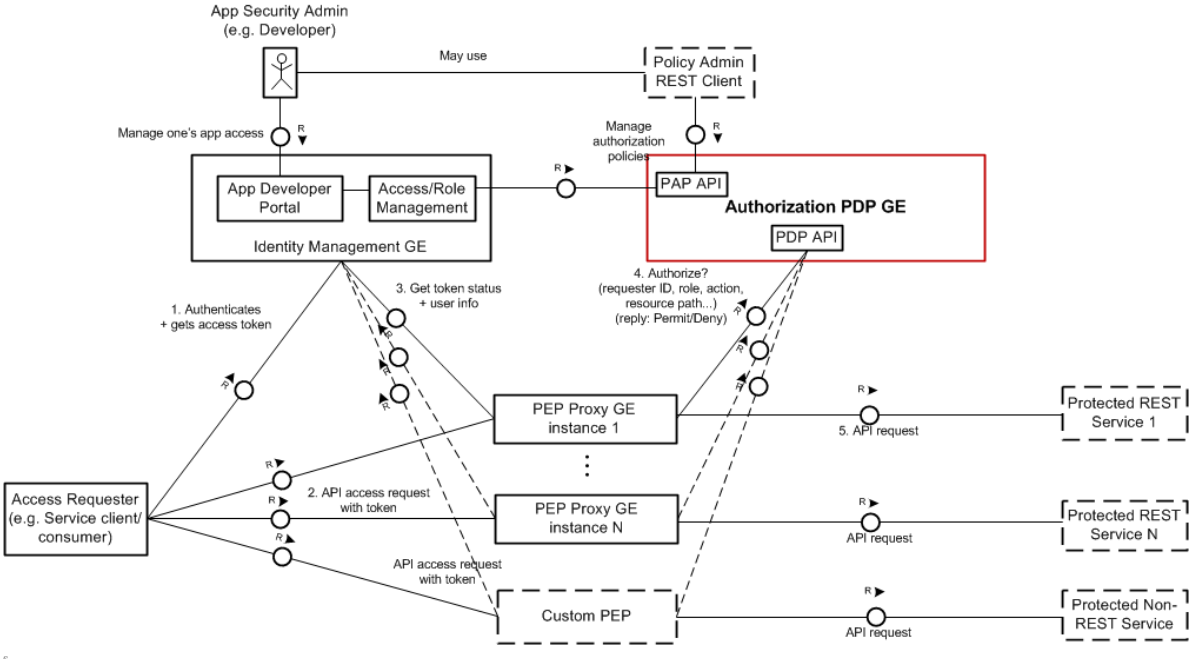
- **Policy language**: XACML defines a XML data model for defining authorization policies, as well as the logic to follow to evaluate them in a given access request context. *Rule* , *Policy* (set of *Rules*), and *PolicySet* (set of *Policy* elements) constitute the main elements of the model. In short, a rule consists of a condition on the access request attributes, and a decision – *Permit* or *Deny* - to apply if the condition holds true for the request. A *Policy* (resp. *PolicySet*) combines multiple *Rules* (resp. *Policies*) and therefore multiple decisions together in various ways (defined in the standard) to make the final decision.
- **Request-Response protocol**: The XACML standard also defines a XML data model for the authorization decision request (XACML *Request*) that a PEP (described later) creates with all the necessary access request attributes and sends to the PDP API for evaluation; and the resulting response (XACML *Response*) that contains the final decision (Permit or Deny).
- **Architecture framework**: The XACML standard also defines a high-level architecture that we are re-using and adapting in FIWARE Security Chapter, including the following major components:
    - **Policy Decision Point (PDP):** The PDP provides authorization decisions based on various attributes given at runtime by PEPs about each incoming access request, and XACML policies that define multiple rules checking whether those attributes (and therefore the access request) satisfy certain conditions. The attributes provided by the PEP (see below) about each access request may be attributes about the request itself: The request URL, the HTTP method; about the requester: The access requester ID, requester role. The PDP may add attributes to the context on its own, such as the current date and time when the requested is received. By replacing all the attribute references in the policy with these input values, PDP is able to evaluate the policy and determine whether the access should be granted.
    - **Policy Administration Point (PAP):** The PAP provides an interface for policy administrators to manage XACML policies to be enforced by the PDP. This endpoint is provided by the Authorization PDP GE as well as a RESTful API interface. The IdM GE also provides a form of graphical interface for the PAP, as part of its access management feature. This feature actually uses the Authorization PDP GE's PAP API as backend.
    - **Policy Enforcement Point (PEP)**: The PEP protects a given resource/service API – typically a REST API in FIWARE – and enforces the decision of the PDP whether to allow or deny a particular access request to the API. The PEP is

worth mentioning here as it is the main component interacting with the PDP to control access to resources.

The PEP can be deployed as a security proxy that intercepts all HTTP(S) traffic to the Resource Server. This kind of PEP is specified in FIWARE as a GE (with associated Reference Implementation): The PEP Proxy GE. Therefore, please refer to the PEP Proxy GE architecture description for more information. In some more complex use cases, e.g. with non-web services, it is not possible to delegate the PEP function to the PEP Proxy GE; it is better to develop a custom one, therefore the *Custom PEP* shown in the diagram of the Overview section.

4.6 Main Interactions

The main interactions are illustrated in the figure below, already shown in the overview but reminded here as we go into further details on the interactions with the GE and other components in FIWARE.



Authorization PDP - Main Interactions

- **Interactions with the GE's PAP API** : The RESTful PAP API is used by the Identity Management GE to have the access policies – defined by developers for their applications via the Developer Portal (graphical form of PAP) – pushed to the PDP GE in order to be enforced by the PDP. The IdM may also use it to store and retrieve the policies managed via the portal. The PAP API may also be used directly by any policy administration client (or any REST client whatsoever) that has been developed specifically for some use case, to address specific aspects of access control policy that the IdM GE's developer portal cannot address. For example, the portal does not address the full complexity and expressiveness of XACML, and it is not its goal. But it is likely that some use cases will need to define policies that require specific features

of XACML (functions, combining algorithms ...) that are not supported by the IdM GE's portal.

The PAP API supports the usual HTTP methods (GET, PUT...) to create, read, update and delete policies. The data representation type is XML. The main type of XML element used in requests/responses is the XACML *PolicySet* as defined in the XACML schema.

- **Interactions with the GE's PDP API** : This REST API is used by PEPs such as the PEP Proxy GE for REST services, or a Custom PEP for other types of service (depending on the use case requirements or constraints), to request an authorization decision from the policy evaluation engine. The PEP sends a HTTP POST request to the API with a XACML Request (defined by the XACML schema) as body of the request. This request should contain all the necessary authorization attributes, mentioned in the policy enforced, for the PDP to be able to evaluate this policy. The PDP returns a XACML Response (defined by the XACML schema) that contains the Permit or Deny decision. It may also return *Indeterminate* if an error occurred during evaluation, for example, if some attribute was missing (not provided by the PEP). You can also define the policy in such a way that the PDP returns *Deny* instead of *Indeterminate* if such an error occurs, to avoid any issue on the PEP side.

4.7 Basic Design Principles

- The architecture complies with the XACML standard.
- All APIs (XACML Policy Administration API, XACML PDP API) are RESTful APIs.
- The format of access control policies managed by the PAP and enforced by the PDP is defined by the XACML standard.
- The PDP computes access decisions based on policies according to the XACML standard rules for policy evaluation (XACML engine).
- The PDP and PAP should support multi-tenancy.

4.8 References

- OAuth 2.0 Authorization Framework
- OASIS XACML 3.0

4.9 Detailed Specifications

Please refer to FIWARE Authorization PDP API Specification for a detailed specification of the Authorization PDP Generic Enabler's interfaces.

4.10 Re-utilised Technologies/Specifications

The GE specification is based on the following:

- OASIS - eXtensible Access Control Markup Language (XACML) ,
- W3C - Web Application Description Language (WADL)
- IETF - RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace.

4.11 Terms and definitions

This section comprises a summary of terms and definitions introduced during the previous sections. It intends to establish a vocabulary that will be help to carry out discussions internally and with third parties (e.g., Use Case projects in the EU FP7 Future Internet PPP). For a summary of terms and definitions managed at overall FIWARE level, please refer to FIWARE Global Terms and Definitions

- **Access control**: is *the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner* ((ITU-T Recommendation X.800). More precisely, access control is the protection of resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entities according to that policy (RFC 2828).
- **XACML**: OASIS standard for eXtensible Access Control Markup Language (XACML 3.0).
- **Policy Decision Point (PDP)**: The PDP provides authorization decisions based on various attributes and XACML policies. Attributes may come from the access request context as provided by PEPs (see below), such as the request URL, the HTTP method and especially the OAuth access token.
- **Policy Enforcement Point (PEP)**: the PEP protects a given resource/service provider's API – typically a REST API in FIWARE - and enforces the decision of the PDP whether to allow or deny a particular access request from a client to the API.

- **Policy Administration Point (PAP)**: the PAP provides an interface for policy administrators to manage XACML policies to be enforced by the PDP.