FI-WARE High-level Description (Product Vision)

FI-WARE-11-08-1

Topics addressed: First official release of the FI-WARE Product Vision

Editor: Juanjo Hierro

e-mail: jhierro@tid.es

Changes History

Release	Major changes description	Date	Editor
1.0	First integrated version of the FI-WARE High-level Description (Product Vision)	11-08-2011	Juanjo Hierro (chief editor)

Use of this Document – Terms, Conditions and Notices

This page will include legal text providing the terms and conditions of use as well as any relevant notices that readers of documents should bear in mind when reading the document

<legal text>

Editing conventions

Styles for bullet and numbered lists

Bullet lists comply with the following conventions:

- An item that is part of a first-level bullet list like the current one use this style.
- This is another item that is part of a first-level list but has sub-items:
  - Sub-items like this one must use the “ItemLevel2” style.
  - This is another sub-item that, in turn, may have sub-items:
    - In which case, they must use the “ItemLevel3” style like this one.
    - Or this another one.
- An item that is part of a first level list may be immediately followed by an indented paragraph like the following one:

  Such paragraph should be indented this way.

Numbered lists are currently allowed only for the first level:

1. This is an item in a numbered list.
2. Followed by another one.

Definition or “points to highlight” sections

Formal definitions or any specific text including points to highlight will be introduced using grey shaded boxes like the following:

- **Term1**. This might be the text defining term1
- **Term2**. This might be the text defining term2

Figures

Figures will be inserted so that number and title of the figure are presented right after the figure.

Tables

Tables will be inserted so that the title precedes the table. Here it is an example:

This is a sample table

Id	Description	Date	Responsible

Scenarios

Description of an scenario or "user story" will be presented using grey shaded boxes like the following:

Scenario 1. Going to the airport

John Davies has to take his flight to Madrid and …

Sample software code

Sample software code may be inserted in white boxes like the following.

```
// comments, of course, should also use the "Software Code" style
int main (argc, argv) {
  …
}
```

Table of Contents

1 INTRODUCTION 8

1.1 ABOUT THIS DOCUMENT 8
1.2 INTENDED AUDIENCE 8
1.3 CONTEXT 8
1.4 STRUCTURE OF THIS DOCUMENT 8
1.5 ASSOCIATED DOCUMENTS 8

2 FI-WARE OVERALL VISION 9

2.1 VISION AND GOALS 9
2.2 BUSINESS ECOSYSTEM 12
2.2.1 Established Telecom Industry 12
2.2.2 IT Industry 13
2.2.3 Usage Areas 13
2.2.4 Emerging Future Internet solution aggregators for converged services 13
2.2.5 End users 14
2.3 SUMMARY OF KEY CONCEPTS INTRODUCED 14

3 CLOUD HOSTING 16

3.1 OVERVIEW 16
3.2 GENERIC ENABLERS 21
3.2.1 IaaS DataCenter Resource Management 21
3.2.2 IaaS Service Management 26
3.2.3 PaaS Management 32
3.2.4 Object Storage 36
3.2.5 IaaS Cloud-edge Resource Management 39
3.2.6 Resource Monitoring 42
3.3 QUESTION MARKS 44
3.3.1 Security aspects 44
3.3.2 Other topics still under discussion 44
3.4 TERMS AND DEFINITIONS 45
3.5 REFERENCES 46

4 DATA/CONTEXT MANAGEMENT 1

4.1 OVERVIEW 1
4.2 GENERIC ENABLERS 5
4.2.1 Publish/Subscribe Broker 5
4.2.2 Complex Event Processing 8
4.2.3 Big Data Analysis 14
4.2.4 Multimedia analysis 18
4.2.5 Unstructured data analysis 24
4.2.6 Meta-data Pre-processing 27
4.2.7 Localization Platform 29
4.2.8 Query Broker 32
4.2.9 Semantic Annotation 36
4.2.10 Semantic Application Support 39
4.3 GENERIC ENABLERS IMPLEMENTING INTELLIGENT SERVICES 41
4.3.1 Social Network Analysis 41
4.3.2 Mobility Analysis 42
4.3.3 Real-time recommendations 44
4.3.4 Web behaviour analysis for profiling 45
4.3.5 Opinion mining 46
4.4 QUESTION MARKS 47
4.4.1 Security aspects 47
4.4.2 Other topics still under discussion 49
4.5 TERMS AND DEFINITIONS 52
4.6 REFERENCES 53

5 APPLICATIONS/SERVICES ECOSYSTEM & DELIVERY 54

5.1 OVERVIEW 54
5.2 GENERIC ENABLERS OF THE BUSINESS FRAMEWORK 58

5.2.1 USDL Service Descriptions....60
5.2.2 Repository....65
5.2.3 Registry....67
5.2.4 Marketplace....68
5.2.5 Business Models & Elements Provisioning System....72
5.2.6 Revenue Settlement & Sharing System....74
5.2.7 SLA Management....77
5.3 GENERIC ENABLERS FOR COMPOSITION AND MASHUP....81
5.3.1 Composition editors....85
5.3.2 Composition execution engines....88
5.4 GENERIC ENABLERS FOR GATEWAY....90
5.4.1 Mediation....90
5.5 GENERIC ENABLERS FOR MULTI-CHANNEL AND MULTI-DEVICE ACCESS....93
5.5.1 Multi-channel/Multi-device Access System....93
5.6 QUESTION MARKS....95
5.6.1 Security Aspects....95
5.6.2 Preliminary Generic Enablers....96
5.6.3 Domain specific extensions for aggregation....96
5.6.4 Relationship to I2ND chapter....97
5.7 TERMS AND DEFINITIONS....97

6 INTERNET OF THINGS (IOT) SERVICES ENABLEMENT....101

6.1 OVERVIEW....101
6.2 GENERIC ENABLERS....104
6.2.1 IoT Communications....106
6.2.2 IoT Resources Management....109
6.2.3 IoT Data handling....113
6.2.4 IoT Process Automation....116
6.3 QUESTION MARKS....118
6.4 TERMS AND DEFINITIONS....119
6.5 REFERENCES....122

7 INTERFACE TO NETWORKS AND DEVICES....127

7.1 OVERVIEW....127
7.1.1 I2ND high-level Architecture....129
7.2 GENERIC ENABLERS....133
7.2.1 Connected Devices Interfacing (CDI)....133
7.2.2 Cloud Edge....142
7.2.3 Network Information and Control (NetIC)....146
7.2.4 Service, Capability, Connectivity, and Control (S3C)....150
7.3 QUESTION MARKS....155
7.3.1 Security aspects....155
7.3.2 Other topics still under discussion....155
7.4 TERMS AND DEFINITIONS....156
7.5 REFERENCES....157

8 SECURITY....158

8.1 OVERVIEW....158
8.2 GENERIC ENABLERS....160
8.2.1 Security monitoring....160
8.2.2 Identity Management....166
8.2.3 PrimeLife Policy Language (PPL) Engine....168
8.2.4 Identity Mixer (IdeMix)....169
8.2.5 Context-based security and compliance....170
8.2.6 Optional Security Service Enabler....173
8.3 QUESTION MARKS....176
8.3.1 Questions still under discussion....176
8.4 TERMS AND DEFINITIONS....178
8.5 REFERENCES....180

1 Introduction

1.1 About this document

This document provides a high-level description of the FI-WARE Platform, easing development of smart applications in the Future Internet.

This document is a living document. The most updated version of the document can be found at:

- http://www.fi-ware.eu/vision

More information about FI-WARE can be found at:

- http://www.fi-ware.eu

1.2 Intended audience

Anyone who wants to get an overall vision of what the FI-WARE project intends to achieve and what are the kind of functionalities that the FI-WARE platform is going to provide to developers of future Internet Applications.

1.3 Context

The FI-WARE platform is the Core Platform whose development is being targeted within the Future Internet PPP initiative launched by the European Commission in collaboration with the Industry. More information about Future Internet PPP initiative can be found at:

- http://www.fi-ppp.eu
- http://ec.europa.eu/information_society/activities/foi/lead/fippp/index_en.htm

The FI-WARE platform is being developed following some of the principles of the Agile methodology. As such, this document corresponds to the FI-WARE Product Vision and is frequently referred as such.

1.4 Structure of this document

A first chapter providing a general overview of the FI-WARE platform is presented. Here, we introduced a number of basic concepts and principles that are general to FI-WARE (Generic Enablers, FI-WARE Instance, etc). These are key to understand the whole vision.

The Reference Architecture of the FI-WARE platform is structured along a number of technical chapters. A description of each of these chapters follows the overall FI-WARE vision. Each chapter is structured so that a general overview of the chapter is provided in the first place. There, main Generic Enablers in the chapters are identified and briefly described. Then, a more detailed description of each of these Generic Enablers is provided. A precise list of terms and definition is provided that can work as basic vocabulary that will help to establish the basis for fruitful discussions around FI-WARE. A list of references is also provided that could be helpful to complement what has been described. Finally, a description of points still under discussion is included. It will transparently present some topics that are currently being targeted as part of ongoing discussions within the project.

1.5 Associated documents

There are no specific documents associated to this one apart from the ones being referred here or in the “References” section accompanying the description of each of the technical chapters on which the FI-WARE Reference Architecture is structured.

2 FI-WARE Overall Vision

2.1 Vision and Goals

The high-level goal of the FI-WARE project is to build the Core Platform of the Future Internet. This Core Platform, also referred to as the "FI-WARE Platform" or simply "FI-WARE" throughout the rest of this document, will dramatically increase the global competitiveness of the European ICT economy by introducing an **innovative infrastructure for cost-effective creation and delivery of versatile digital services, providing high QoS and security guarantees.** As such, it will provide a powerful foundation for the Future Internet, stimulating and cultivating a sustainable ecosystem for (a) innovative service providers delivering new applications and solutions meeting the requirements of established and emerging Usage Areas; and (b) end users and consumers actively participating in content and service consumption and creation. Creation of this ecosystem will strongly influence the deployment of new wireless and wired infrastructures and will promote innovative business models and their acceptance by final users.

FI-WARE will be open, based upon elements (hereunder called Generic Enablers) which offer reusable and commonly shared functions serving a multiplicity of Usage Areas across various sectors**.** Altogether, such a platform will address the challenges described in the previous section. Note that not all functions that are common to applications in a given Usage Area may lead to the identification of Generic Enablers (GEs). It is the ability to serve a multiplicity of Usage Areas that distinguishes Generic Enablers from what would be labelled as Domain-specific Common Enablers (or "Specific Enablers" for short), which are enablers that are common to multiple applications but all of them specific to one particular Usage Area or a very limited set of Usage Areas.

Key goals of the FI-WARE project are the identification and specification of GEs, together with the development and demonstration of reference implementations of identified GEs. Any implementation of a GE comprises a set of components and will offer capabilities and functionalities which can be flexibly customized, used and combined for many different Usage Areas, enabling the development of advanced and innovative Internet applications and services. The FI-WARE Architecture comprises the description of GEs, relations among them and properties of both.

Specifically, the Core Platform to be provided by the FI-WARE project is based on GEs linked to the following main architectural chapters:

- **Service Delivery Framework** – the infrastructure to create, publish, manage and consume FI services across their life cycle, addressing all technical and business aspects.
- **Cloud Hosting** – the fundamental layer which provides the computation, storage and network resources, upon which services are provisioned and managed.
- **Data/Context Management Services** – the facilities for effective accessing, processing, and analyzing massive streams of data, and semantically classifying them into valuable knowledge.
- **IoT Enablement** – the bridge whereby FI services interface and leverage the ubiquity of heterogeneous, resource-constrained devices in the Internet of Things.
- **Interface to the Network and Devices** – open interfaces to networks and devices, providing the connectivity needs of services delivered across the platform.
- **Security** – the mechanisms which ensure that the delivery and usage of services is trustworthy and meets security and privacy requirements.

In order to illustrate the concept of GE, let's analyze some of the GEs that have been initially identified as linked to one of the defined Architecture Chapters in FI-WARE, e.g., the chapter linked to Data/Context Management Services. This chapter may comprise some sort of GE that allows compilation and storage of massive data from disparate sources. Note that this GE may in turn be based on a number of GEs, each specialized in gathering data from a specific source (e.g., data from connected "things", data obtained from user devices, data provided by the user, data exported by applications, etc.), The Context/Data Management

Service may also comprise a number of GEs dealing with processing of stored data, enabling generation/inferencing of new valuable data that applications may be interested to consume. It may finally comprise a GE which supports a well defined API enabling Future Internet Applications to subscribe to data they are interested in, making them capable of receiving this data in real time.

To a large degree, the functionalities of the Generic Enablers will be driven by requirements from Use Case projects in the context of the PPP. Therefore, FI-WARE will closely collaborate with Use Case projects and the Capacity Building project in the FI-PPP program. However, it will also be driven by requirements extrapolated for any other future services. These additional requirements may be brought by partners of the FI-WARE consortia (based on input of their respective Businesses Units) or gathered from third parties external to PPP projects.

The FI-WARE project will introduce a generic and extendible ICT platform for Future Internet services. The platform – also referred to as the "Future Internet Core Platform" or "FI-WARE" – aims to meet the demands of key market stakeholders across many different sectors, strengthen the innovation-enabling capabilities in Europe and overall ensure the long-term success of European companies in a highly dynamic market environment.

Products implementing FI-WARE GEs can be picked and plugged together with complementary products in order to build FI-WARE Instances, operated by so called FI-WARE Instance Providers. Complementary products allow FI-WARE Instance Providers to differentiate their offerings and implement their desired business models. For example, a company playing the FI-WARE Instance Provider role may decide to develop and/or integrate their own set of monitoring/management tools, because this will allow it to benefit from a better integration with the tools already used by the company for the monitoring/managing of other services, therefore making operations much more efficient. FI-WARE Instance Providers would also typically develop their own solutions for monetization of the services delivered through the FI-WARE Instance they operate. This solution will typically imply integration with proprietary Billing or Advertising Support Systems. In this respect, GEs defined in FI-WARE will not impose restrictions on the particular business model a FI-WARE Instance Provider intends to implement.

FI-WARE GEs are further classified into core FI-WARE GEs and optional FI-WARE GEs. Core FI-WARE GEs are required to be deployed in every FI-WARE Instance.

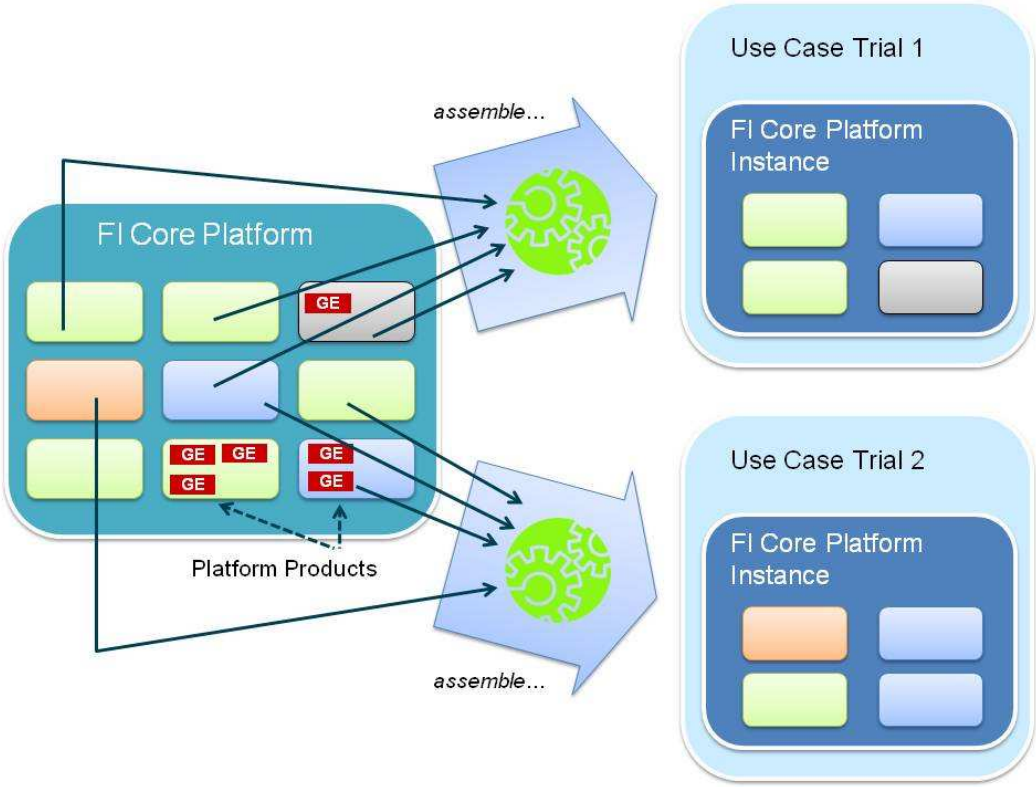


Figure 1 FI-WARE Instances

The set of strategic goals of the FI-WARE project are as follows:

- To specify, design, and develop a Core Platform (hereunder referred to as "FI-WARE") to be a generic, flexible, trustworthy and scalable foundation, supporting the equations listed previously.
- Design extension mechanisms so as to enable to support for yet unforeseen Usage Areas not being addressed in the context of the FI-PPP. This requires a suitable extrapolation of current technology and business trends and their translation into the specific design and implementation principles of FI-WARE.
- To liaise between the project and relevant standardisation bodies in order to: a) to keep the project up-to-date with respect to the discussions in the standardisation bodies; b) to support the submission of contributions from the project in a coordinated way. The aim is to ensure active contribution of specifications leading to open standardised interfaces.
- To implement and validate the FI-WARE approach in trials together with Use Case projects in order to develop confidence for large scale investments in solutions for smart future infrastructures on national and European level.
- To enable established players (telecoms, etc.) and emerging players in the services and application domains to tap into new business models by providing components, services, and platforms for these emerging players to innovate.
- To support the development of a new ecosystem including agile and innovative service providers consuming components and services from FI-WARE thereby building new business models based on FI-WARE and associated Usage Areas.
- To stimulate early market take-up by promoting project results jointly with the other projects in the FI-PPP.

In the FI-WARE project, there is an equal focus on advancing components/technologies linked to Generic Enablers as well as integrating them in order to meet the actual requirements of all stakeholders. R&D activities in FI-WARE will comprise:

- Evolving components contributed by partners or available on the market in order to:
  - incorporate new features not covered by these components but required to implement GEs in the context of the Future Internet
  - allow GEs implemented through these components to be integrated (pluggable) with other GEs in the FI-WARE Architecture
- Creating new components that may cover gaps in the FI-WARE Architecture

The FI-WARE project will draw upon the wealth of results already achieved through earlier European research, not only within the FP's but also at national- or corporate-funded levels, and to leverage them further through a systematic integration with a complete system perspective.

2.2 Business Ecosystem

The following table describe the different roles in the overall value chain defined around FI-WARE:

Roles in the value chain defined around FI-WARE

Role	Description
FI-WARE GE Provider	Any implementer of a FI-WARE GE. The open and royalty-free nature of FI-WARE GE specifications will allow parties other than partners in the FI-WARE consortium to develop and commercialize products that are in compliance with FI-WARE GE specifications.
FI-WARE Instance Provider	A company or organization which deploys and operates a FI-WARE Instance and establishes some sort of business model around that particular FI-WARE Instance. Note that in building a FI-WARE Instance, a FI-WARE Instance Provider may rely on products being developed in the FI-WARE project or products being developed by any sort of FI-WARE GE Provider. Also note that a FI-WARE Instance Provider may decide to be the Provider (implementer) of some of the GEs in the FI-WARE Instance they operate.
FI-WARE Application/Service Provider	A company or organization which develops FI applications and/or services based on FI-WARE GE APIs and deploys those applications/services on top of FI-WARE Instances. Note that the open nature of FI-WARE GE specifications enables portability of FI applications and/or services across different FI-WARE Instances.

Many different business stakeholders will be part of the ecosystems creating Future Internet based applications and services. They are likely to have different business objectives and offerings and therefore will take one or more of the roles defined above. The following categorisation summarises the relevant stakeholders, the roles they are expected to play and, consequently, the impact FI-WARE may induce:

2.2.1 Established Telecom Industry

- Telecom Operators: They will typically play the role of FI-WARE Instance Providers, relying on FI-WARE technologies to develop new business models for leveraging the assets they already have, i.e. exploring possibilities to "open up" their existing networks and generating new revenue streams (e.g.

by appropriate IaaS / NaaS models). They will need to understand the technical hurdles, which need to be overcome. They can also play a relevant role accelerating the development of standards based on FI-WARE results.

- Network equipment manufacturers (core and access networks): Develop and provide technical solutions to support Telecom Operators in their role as FI-WARE Instance Providers. They may generate product/platforms and provide services on which Telecom Operators may rely to implement their role as FI-WARE Instance Providers. They may use existing and adapted assets in new Usage Areas for such technical solutions. They can also play the role of FI-WARE Application/Service Providers, commercializing compelling ICT-based services based on FI-WARE which can be bundled together with other Application/Services in their portfolio to bring solutions to different Usage Areas.
- Mobile terminal manufacturers: Develop and provide appropriate terminal devices with new features, M2M communication equipment and sensors for new application domains, which can interface easily with FI-WARE Applications and Services. Based on open standardized interfaces, economies of scale and affordable/interchangeable solutions for customers can be achieved.
- Telecom solution providers (note that some Telecom Operators are also Telecom Solution Providers): They will typically play the role of FI-WARE Application/Service Providers. They develop new services and applications for large Usage Areas, where telecom-based communication, security and availability levels as well as support to roaming users are required.

2.2.2 IT Industry

- Software vendors: They will typically play the role of FI-WARE GE Provider. Therefore, they will explore, develop and provide new software components, services, middleware, business services (delivery) platforms and cloud-based infrastructure components needed for emerging Future Internet applications and services in a converging IoT, IoS, and IoC environment.
- Service providers: They will typically play the role of FI-WARE Application and Service Providers. Therefore, they will deal with provisioning and operation of innovative applications and services on top of FI-WARE Instances. In some cases, they may also play the role of FI-WARE Instance Providers, deploying FI-WARE Instances dedicated to run the Applications and Services they have developed (e.g. following a sort of ASP model).
- IT Solution providers: They will typically play the role of FI-WARE Application and Service Providers, adapting to the new challenges of developing and integrating new converged Telecom/IT solutions. Similarly to Service Providers, they may also play the role of FI-WARE Instance Providers, deploying FI-WARE Instances dedicated to run the Application and Services they have developed.

2.2.3 Usage Areas

The various stakeholders in the Usage Areas are expected to have very different objectives. Many of them suggest basic improvements of their business processes and the more efficient use of resources of any kind. Others request solutions to support them in the development of new cross-sector solutions currently considered as complex to implement and operate. Both of them are in charge of supporting societal challenges either based on financial incentives or based on regulatory requirements. FI-WARE is a key element to support the different sectors in their approach.

2.2.4 Emerging Future Internet solution aggregators for converged services

In a converged Future Internet ICT industry new challenges for developing / composing / deploying of (domain- or sector-specific) solutions emerge. Established or new players especially from the SME sector will increasingly have to develop solutions in a world of complex service networks crossing telecom

services. Therefore, they will play the role of FI-WARE Application and Service Providers, thereby adapting their business models and technology know-how appropriately.

Note that convergence equally well applies to cross-sector innovation concerning domain-specific service providers, e.g. from the areas of logistics, healthcare, energy, services, where new services are composed by linking and services from different sectors and developing new business models around them.

2.2.5 End users

Further end users affected by the FI-PPP and contributions of FI-WARE are citizens, (non-governmental) organisations, individuals, employees, and the generation of prosumers (possibly also aiming at generating their personal income). Solutions that have direct impact on this group of stakeholders are highly desirable.

2.3 Summary of key concepts introduced

Following are the definitions of some terms that are key to the FI-WARE vision and are widely used throughout the entire document. As mentioned before, the terms “Core Platform”, “CP” or “FI-WARE” can be used indistinctly throughout the document:

- **FI-WARE Generic Enabler (GE)**: A functional building block of FI-WARE. Any implementation of a Generic Enabler (GE) is made up of a set of components which together supports a concrete set of Functions and provides a concrete set of APIs and interoperable interfaces that are in compliance with open specifications[1] published for that GE.
- **FI-WARE compliant Platform Product**: A product which implements, totally or in part, a FI-WARE GE or composition of FI-WARE GEs (therefore, implements a number of FI-WARE Services). Different FI-WARE compliant Platform Products may exist implementing the same FI-WARE GE or

[1] GE Open Specifications will contain all the information required in order to build compliant products which can work as alternative implementations of GEs developed in FI-WARE and therefore may replace a GE implementation developed in FI-WARE within a particular FI-WARE Instance. GE Open Specifications will typically include, but not necessarily will be limited to, information such as:

- Description of the scope, exhibited behaviour and intended use of the GE
- Terminology, definitions and abbreviations to clarify the meanings of the specification
- 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.
- Description of protocols that support interoperability with other GE or third party products
- Description of non-functional features

composition of FI-WARE GEs. Actually, the open and royalty-free nature of FI-WARE GE specifications allows the existence of alternative implementations of a FI-WARE GE. FI-WARE compliant Platform Products are made up of components. While implementations of Generic Enablers developed in compliance with FI-WARE GE Open Specifications are replaceable, components linked to a particular FI-WARE compliant Platform Product may not be replaceable.

- **FI-WARE Instance**: The result of the integration of a number of FI-WARE compliant Platform Products and, typically, a number of complementary products. As such, it comprises a number of FI-WARE GEs and supports a number of FI-WARE Services. Provision of Infrastructure as a Service (IaaS) or Context/Data Management Services are examples of FI-WARE Services a particular FI-WARE Instance may support, implemented by means of combining a concrete set of Platform Products. While specifications of FI-WARE GEs define FI-WARE in functional terms, FI-WARE Instances are built by means of integrating a concrete set of FI-WARE compliant Platform Products.
- **FI-WARE Instance Provider**: A company that operates a FI-WARE Instance. Note that FI-WARE Instances may not consist only of the integration of FI-WARE compliant Platform Products but their integration with other products which allow the FI-WARE Instance Provider to gain differentiation on the market (e.g. integration with own Operating Support Systems to enhance FI-WARE Instance operation or with other products supporting services that are complementary to those provided by FI-WARE GEs) or to enable monetization of its operation (e.g., integration with own Billing or Advertising systems).
- **Future Internet Application**: An application that is based on APIs defined as part of GE Open Specifications. A Future Internet Application should be portable across different FI-WARE Instances that implement the GEs that Future Internet Application relies on, no matter if they are linked to different FI-WARE Instance Providers.
- **FI-WARE Testbed**: A concrete FI-WARE Instance operated by partners of the FI-WARE project that is offered to Use Case projects within the FI-PPP Program, enabling them to test their proof-of-concept prototypes. The FI-WARE Testbed is also offered to third parties to test their Future Internet Applications although support to them is provided on best-effort basis.
- **FI-WARE Instance in production**: A FI-WARE Instance run by a FI-WARE Instance Provider in the context of a trial (e.g., trials in phase 2 of the FI PPP) or as part of its service offering to the market. FI-WARE Instances in production will typically have their own certification and developers community support environments. However, several FI-WARE Instance Providers may establish alliances to setup common certification or developers community support environments.

3 Cloud Hosting

3.1 Overview

Cloud computing is nowadays a reality. Cloud hosting companies, which can be considered as a particular type of FI-WARE Instance Providers, are already delivering on the promise of the Cloud paradigm. They own and manage large IT infrastructures and offer their use as a service on a pay-as-you-go model.

Cloud hosting is particularly appealing to SMEs and start-ups wanting to offer some new and innovative service over the Internet. Actually, it offers SMEs general purpose computing resources that they can consume (and pay) according to their needs and capabilities, e.g. they can start small and grow as the service they offer becomes successful. All this is achievable without the need for large initial investment in the infrastructure. This in turn gives the SMEs a possibility to competitively price their offerings since there is no need to recover a huge initial capital investment in infrastructure and, in addition, the on-going operational expenses are lowered thanks to the pay-as-you-go model.

Today, there are two clear trends in the cloud computing market: 1) growing adoption of the full cloud computing paradigm, as exemplified by public clouds; and, 2) the appearance of private clouds, i.e., the adoption of the cloud ideas and technologies internally within companies. The latter approach is especially appealing for large companies that are already operating large data center infrastructures. On one hand, they are still reluctant to fully adopt the cloud hosting model and rely solely on external providers for their IT needs (due to various factors such as security and privacy as well as performance and availability guarantees). On the other hand they do want to benefit from advantages that cloud computing paradigm introduces in terms of cost and flexibility. Such a trade-off also introduces a hybrid approach where private clouds incorporate facilities to burst workload on public clouds (cloudbursting), This approach is not only fundamental for large companies but is increasingly gaining momentum among SMEs who need to gain the necessary confidence on the Cloud promise prior the full outsourcing of their computing infrastructures.

However, as the IT infrastructure moves from being owned and managed by the service providers to being hosted on the cloud, the cloud hosting companies become a critical part of their customers' businesses This creates a dependency relationship that could even lead to unhealthy and undesirable situations such as vendor lock-in, if the necessary safeguards in terms of technology, market offerings and warranties are not in place.

Moreover, the cloud hosting market is still limited to a few, very dominant, large companies with proprietary solutions. The lack of a competitive and open market for cloud hosting providers, in turn, slows down the adoption of the cloud paradigm and the economic benefits embodied in it. For the success of the Internet-based service economy it is crucial that cloud hosting does not become a market limited to a few strong players, and that future cloud hosting is based on open standards and support interoperability and portability.

The FI-WARE project focuses a great part of its efforts on making sure that these standards materialise and in facilitating their adoption by providing open specifications and reference implementations. This standards-based and open approach will cover the fundamental technologies on which the cloud paradigm is based, such a virtualization, as well as new emerging technologies that will differentiate FI-WARE also in terms of the functionality offered.

In the cloud hosting paradigm there are two main players: (1) cloud hosting providers, i.e., FI-WARE Instance Providers that own physical infrastructure and use it to host compute processes or applications; and (2) cloud hosting users, i.e., organizations or individuals that own the compute processes or applications but do not own (or do not want to own) the physical infrastructure to run them, hence, they lease this infrastructure from cloud hosting providers. In a highly distributed cloud model, the physical infrastructure is deployed very close to the end-user and can be controlled by the network and platform provider or the end-user directly.

According to the needs of clients there are three well-defined Cloud Service offerings [REF:NIST]:

- Infrastructure as a Service (IaaS): in this model the client rents raw compute resources such as storage, servers or network, or some combination of them. These resources become available on the Internet (public IaaS Clouds) or on the Intranets (private IaaS Cloud) with capacities (storage space, CPUs, bandwith) that scale up or down adapted to real demand of applications that uses them. The advantage of this model is that enables users to setup their own personalized runtime system architecture. However it allows this at the price of still requiring system admin skills by the user..
- Platform as a Service (PaaS): in this model the clients, typically application developers, follow a specific programming model and a standard set of technologies to develop applications and/or application components and then deploy them in a virtual application container or set of virtual application containers they rent. How these virtual application containers map into a concrete runtime architecture is hidden to the application develper who doesn't need to have strong admin skills. However, this happens at the price of loosing part of the control on how the system runtime architecture is designed. This model enables fast development and deployment of new applications and components. Usually, in addition to hosting, the PaaS providers also offer programming utilities and libraries that expedite development and encourage reuse.
- Software as a Service (SaaS): in this model the client, typically end-users, rent the use of a particular hosted Final Application, e.g., word processing or CRM without needing to install and executed the application on top of equipments owned by the client (consumers) or assigned to the client (employees in a company). Applications delivered following a SaaS model are always available so that clients get rid of maintenance tasks (including upgrading, configuration and management of high-availability and security aspects, etc). Computing resources needed to run applications on owned/assigned clients get minimized since they are hosted on the Internet/Intranet.

While it is possible to implement these models independently of each other, i.e., SaaS without PaaS or IaaS, PaaS without IaaS, the advantages offered by each of these models to its potential users are such that we strongly believe that the vast majority of the Future Internet services will be based on a stacked implementation of these models as shown in The XaaS Stacked Model Figure below.

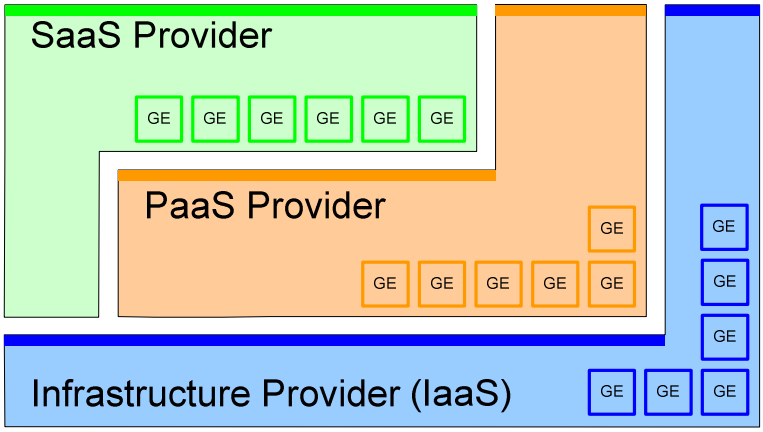


Figure 1 The XaaS stacked model

In our vision, Application Providers willing to offer applications following a SaaS model will typically opt to implement this model using the services of a PaaS or a IaaS Cloud provider. Usage of a PaaS Cloud provider will mostly apply to Application Providers who a) have decided to adopt a defined standard platform for the development of applications and b) wish to focus their skills in programming and application architectural aspects without needing to hire experts who can deal with the design and fine tuning of large system runtime architectures. However, Application Providers may have some special needs that are not properly covered by the PaaS programming model and tools, or wish to be able to design and

configure the system runtime architecture linked to their applications, based on raw computing resources from an IaaS provider. Similarly, PaaS providers may rely on IaaS providers for leasing infrastructure resources on demand. In this context a cloud hosting provider may serve the role of a PaaS Cloud provider, or the role of an IaaS Cloud provider, or both.

The Cloud Hosting chapter in the FI-WARE Reference Architecture will comprise the Generic Enablers that can serve the needs of companies that may need IaaS Cloud hosting capabilities, PaaS Cloud hosting capabilities or both, meeting the requirements for the provision of a cost-efficient, fast, reliable, and secure computing infrastructure "as a Service".

The basic principle to achieve a cost-efficient infrastructure is the ability to share the physical resources among the different users, but sharing needs to be done in a way that ensures isolation (access, control and performance) between these users. These seemingly contradictory requirements can be met by an extensive use of virtualisation technology.

Virtualization capabilities are the cornerstone of any IaaS Cloud Hosting offering because they enable both high utilization and secure sharing of physical resources, and create a very flexible environment where logical computation processes are separated and independent from the physical infrastructure. FI-WARE's base functionalities will include a virtualization layer that will enable secure sharing of physical resources through partitioning, support migration without limitations, and provide a holistic system-wide view and control of the infrastructure. Basic management of the resulting virtualised infrastructure will automate the lifecycle of any type of resource by providing dynamic provisioning and de-provisioning of physical resources, pool management, provisioning, migration and de-provisioning of virtual resources, on-going management of virtual capacity, monitoring etc.

Virtualisation technologies, such as hypervisors or OS containers, enable partitioning of a physical resource into virtual resources that are functionally equivalent to the physical resource. Moreover, virtualisation creates a very flexible environment in which logical functions are separated from the physical resources. IaaS Cloud hosting providers can leverage this capability to further enhance their business. For example live-migration of virtual resources, i.e., the capability of moving the virtual resource from one physical resource to another while the virtual resource remains functional; enable the cloud hosting providers to optimize the resource utilization. However, running different workloads on a shared infrastructure, hosted by a 3rd party, introduces new challenges related to security and trust. FI-WARE will address these challenges by leveraging generic enablers defined in the FI-WARE Security chapter.

In addition to virtualisation and the management of it, cloud hosting providers need a layer of generic enablers that deal with the business aspects of optimally running their operation. Existing IaaS Cloud Hosting technologies and commercial offerings represent a big step forward in terms of facilitating management of compute infrastructure by completely virtualising the physical resources used by software, but still do not fully address all the needs of both IaaS Cloud Hosting Providers and Application and Service Providers. IaaS Cloud Hosting Providers need grouping and elasticity, policy-driven data centre optimisation and placement, billing and accounting, more control over virtualised Network Resources. Application and Service Providers need the infrastructure management decisions to be directly driven by Service Level indicators and not compute parameters as is the case today.

Typically existing IaaS Cloud Hosting solutions are based on a centralised infrastructure deployed usually on a few data centres distributed geographically. However, some Future Internet applications may require reduced latency and high bandwidth that this approach and current network realities cannot always meet. This becomes especially problematic when the users of the hosted applications and services are using their home broadband connections. Stricter privacy requirements that favour local-only storage of data may be an additional obstacle to the current approach, as it would place data even further away from the computational infrastructure. To address these challenges, FI-WARE will explore the possibility to extend the reach of the IaaS Cloud Hosting infrastructure to the edge of the networks by incorporating a device located at the home of an end user, the Cloud Proxy that can host part of the virtualised resources, applications and data, thereby keeping data closer to the user.

Application Providers may rent from IaaS Cloud providers dynamic infrastructure resources to deploy service components, but they are on their own in terms of coming up with the deployment architecture,

managing and deploying enabling SW components, managing and maintaining the software stacks installed on each virtual machine and controlling the scalability of the virtualised infrastructure resources. FI-WARE will build on top of robust virtualisation-based IaaS technologies to create a Platform as a Service offering that provides a higher level of abstraction for service provisioning where the platform itself provides development tools, application containers, integrated technologies (libraries, APIs, utilities, etc.) and automatic scalability tools, allowing the Application Providers to deploy applications by means of providing just the description of their Application Components. The delivery of standard interfaces and reference implementations for the above elements are both in the scope of the FI-WARE.

In order to simplify management of hosted resources FI-WARE will provide a self-service portal where Application and Service Providers will be able to select, configure, deploy and monitor their whole applications and services through graphical tools. Application Blueprints and Service Level Agreements will be used by Cloud Hosting providers to drive automatic provisioning and dynamic management of the virtualized resources.

Trust and consequentially security concerns are one of the top obstacles that hinder Cloud Computing adoption today. FI-WARE will work towards embedding security, privacy and isolation warranties, which can be achieved through use of standard security techniques (authentication, authorization, encryption, etc) and partitioning technologies that warranty isolation, to all layers of its Cloud Hosting platform.

Cloud Hosting will be an integral part of the FI-WARE platform and together with the Apps/Services Ecosystem, Data/Context Management Services, Internet of Things Service Enable and Interfaces to the Network and Devices will offer a complete solution for: application development that automatically resolves hosting, deployment and scalability, provides the necessary interfaces and services so that applications can leverage the Internet of Things, provide intelligent connectivity all through the stack to guarantee QoS, resolve common needs like data storage and analysis, access to context and monetization, allow the delivery of applications through a rich ecosystem that enables the implementation of flexible business models and allows user driven process creation and personalization.

Summarizing the above:

Building upon existing virtualization technologies, FI-WARE will deliver a next generation Cloud Stack that will be open, scalable, resilient, standardised, and secure, and will enable Future Internet applications by providing service-driven IaaS and PaaS functionalities and extending the reach of the cloud infrastructure to the edge of the networks, much closer toendl users.

To better illustrate the FI-WARE proposition for Cloud Hosting let us take a look at a typical scenario for a cloud hosting company.

A start-up company has an idea for an innovative application to be offered as Software as Service (SaaS). They can calculate pretty accurately how many servers and how much storage they will need to support a target number of end-users of their application, but giving that this is a totally new application they cannot estimate whether they will reach this number or surpass it. To reduce the risk involved with a big initial investment in equipment, they decide to lease resources on demand from a cloud hosting provider. They require of the cloud hosting provider to offer unlimited, on-demand and automated growth, i.e., they will start with a minimum set of resources, and the provider commits to automatically add more resources when the service reaches a particular load, and when the load decreases, the additional resources will be released again automatically. They also need from the cloud hosting provider isolation and availability guarantees and they want the flexibility to painlessly switch providers in case of breach of contract or a provider going out of business. Finally they would prefer a provider that can give them raw compute resources for those tasks unique to their application, and also supports composition and hosting of commonly used application components, for example a web-based user interface to their application.

FI-WARE offers to Cloud Hosting companies the tools needed to answer these requirements from their potential customers, in this case the start-up company in need of a flexible on-demand infrastructure.

Figure 2 illustrates the Reference Architecture for the Cloud Hosting chapter in FI-WARE, each box representing one of the Generic Enablers (GEs), which would be part of it.

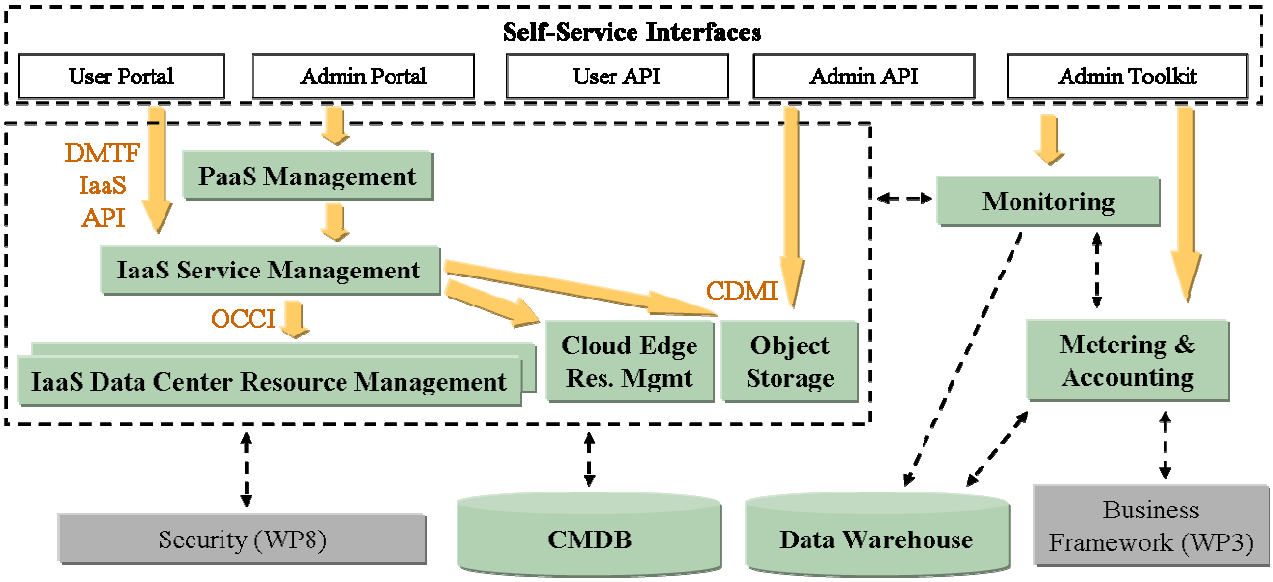


Figure 2 Cloud Hosting Reference Architecture

Herein we provide a brief description of the role each GE plays and their major interfaces with other GEs:

- **IaaS Data Center Resource Management** – this GE provides VM hosting capabilities to the user, and handles everything related to individual VMs and their resources – including compute, memory, network and block storage. This includes provisioning and life cycle management, capacity management and admission control, resource allocation and QoS management, placement optimization, etc.
- **IaaS Cloud-Edge Resource Management** – this GE allows the application developer to design and deploy the application so that it can leverage resources located at the cloud edge, close to the end-user.
- **IaaS Service Management** – this GE provides hosting of compound VM-based services, including their definition and composition, deployment and life cycle management, as well as monitoring and elasticity. This GE uses the IaaS Resource Management GE to handle individual VMs and their resources. This GE is also able to communicate with other clouds, in scenarios of cloud federations.
- **PaaS Management** – this GE provides hosting of application containers, such as Web container, database instance, etc. It leverages IaaS underneath, to automate the lifecycle of the underlying infrastructure and OS stack.
- **Object Storage** – this GE provides the capabilities to store and retrieve storage objects accompanied by metadata.
- **Monitoring** – this GE will be responsible for collecting metrics and usage data of the various resources in the cloud.
- **CMDB** – this GE will be responsible for storing the operational configuration of the Cloud environment, used by the various other GEs. Due to scalability requirements, it is likely to be implemented as a distributed service.
- **Data Warehouse** – this GE will be responsible for storing the historical data of the different metrics and resource usage data of the Cloud environment, collected by the monitoring & metering GE and consumed by the SLO management GE (to monitor SLO compliance), as well as by the billing GE.
- **Metering & Accounting** – this GE will be responsible for collecting and processing the data related to usage and monetization of cloud services (via an external Billing system, which is not part of this GE).

Each GE is described in more detail in Section 3.2 debajo de.

Last but not least, there are two main users of Cloud Hosting GEs:

- **Cloud hosting provider**: uses the provided capabilities to build a hosting offering, and to perform ongoing administration tasks
- **Cloud hosting user**: e.g., a Application/Service providers who uses the provided platform to develop and/or test and/or deploy their applications.

Self-service interfaces will be provided so that different types of users would be able to interact with the entire the FI-WARE cloud infrastructure in a common but unified fashion. It should adapt to different user mental models in order that it is easy to use. Applying techniques common in "Web 2.0" can also help to make it more usable. It is foreseen that there will be different kinds of users with different levels of expertise and adaptation to their expectations and needs should be a goal.

Our objective is that using this infrastructure is a positive, simple and easy experience for all the FI-WARE and other Future Internet users. This will be a key requirement regarding self-service interfaces. Different users have varying requirements in how the interact with Information Technology devices and services. As a result we foresee different types of support to satisfy these requirements. Amongst those are:

- A portal,
- A high-level toolkit that may be integrated with management or development tools and
- Scripts to automate the task are required.

Direct access to the underlying APIs will also be offered should the support listed above be insufficient.

3.2 Generic Enablers

3.2.1 IaaS DataCenter Resource Management

3.2.1.1 *Target usage*

The IaaS DataCenter Resource Management GE provides the basic Virtual Machine (VM) hosting capabilities, as well as management of the corresponding resources within the DataCenter that hosts a particular FI-WARE Cloud Instance.

The main capabilities provided for a **cloud hosting user** are:

- Browse VM template catalogue and provision a VM with a specified virtual machine image
- Manage life cycle of the provisioned VM
- Manage network and storage of the VM
- Resource monitoring of the VM
- Resiliency of the persistent data associated with the VM
- Manage resource allocation (with guarantees) for individual VMs and groups of VMs
- Secure access to the VM

For a **cloud hosting provider**, the following capabilities are provided:

- Resource optimization and over-commit (aimed at increasing the utilization and decreasing the hardware cost)
- Capacity management and admission control (allowing to easily monitor and control the capacity and the utilization of the infrastructure)
- Multi-tenancy (support isolation between VMs of different accounts)
- Automation of typical admin tasks (aimed at decreasing the admin cost)
- Resiliency of the infrastructure and of the management stack (aimed at reducing outage due to hardware failures)

3.2.1.2 *GE description*

In order to achieve **scalability**, the infrastructure is managed in a hierarchical manner, as shown in Figure 3 At the top, the DataCenter-wide Resource Manager (DCRM) is responsible for surfacing the functions and capabilities required for the provision and life-cycle management of VMs and associated resources, as specified above. At the bottom, the Node Manager is responsible for managing the resources provided by individual physical nodes. In between, a number of System Pools may be defined, typically encapsulating homogenous and physically co-located pools of resources (compute, storage, and network). Each system pool has some self-management capabilities, provided by System Pool Resource Manager (SPRM) which exposes to DCRM an abstracted view of its resources, as if it was a 'mega-node', while delegating the operations on individual resources to the next-level SPRM (if there are multiple levels of System Pools), or to the corresponding Node Manager.

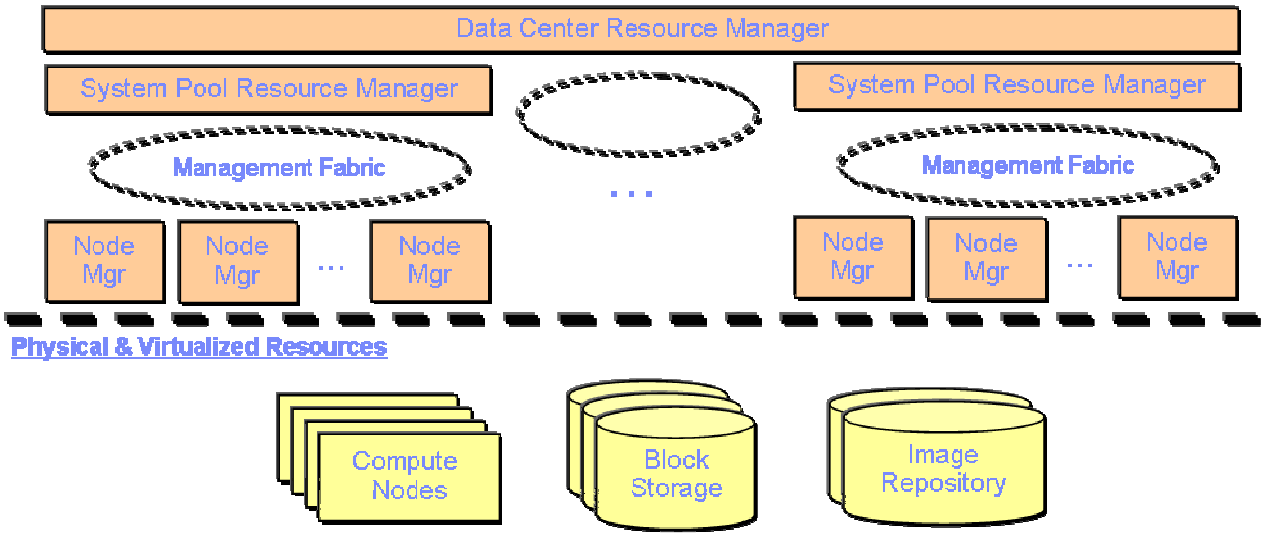


Figure 3 GE Architecture: Data Center Resource Management

Across the three management layers (node, pool, data center) and three resource types (compute, storage, network), the following resource management functions and capabilities are provided:

- Request orchestration and dispatching
- Discovery & inventory
- Provisioning and life cycle management
- Capacity management & admission control
- Placement optimization

- QoS management & resource allocation guarantees
- Resource reservation and over-commit
- Monitoring & metering
- Isolation and security
- Resiliency

For flexibility, the RMs at the different levels of the hierarchy will use **unified interfaces** to communicate between them. This interface will include a core resource model which can be extended for the needs of each resource type to be managed. Each RM could publish a **profile** that details its specific resource management capabilities. Given the importance of interoperability, especially within the context of Generic Enablers, the API and related model should aim to be based on existing open, IPR-unencumbered work. The Open Cloud Computing Interface (OCCI) will be used in FI-WARE for this purpose (see below).

For **resiliency**, the individual RMs will share a common group communication fabric, enabling efficient messaging as well as high availability and fault tolerance of the individual management components (by implementing heartbeat and failover to a standby node).

One of the unique capabilities that FI-WARE is aimed at is providing **resources allocation guarantees** specified via Resource Allocation Service Level Objectives (**RA-SLO**s), enforced by this GE. See more details below.

In order to achieve high **resource utilization**, the RMs will apply intelligent placement optimization and resource over-commit. This task is especially challenging when applied in conjunction with support for performance and RA-SLOs (mentioned above), and requires significant innovation.

Open Cloud Computing Interface

OCCI is a RESTful protocol and API for the management of cloud service resources. It comprises a set of open community-lead specifications delivered through the Open Grid Forum. OCCI was originally initiated to create a remote management API for IaaS model based Services. It has since evolved into a flexible API with a strong focus on integration, portability, interoperability and innovation while still offering a high degree of extensibility.

OCCI aims to leverage existing SDO specifications and integrate those such that where a OCCI specified feature may not be rich enough a more capable one can be brought into play. An excellent example of this is the integration of both CDMI and OVF. In particular to those 2 previously mentioned standards, when combined together provide a profile for open and interoperable infrastructural cloud services [OCCI_OVF_CDMI].

The main design foci of OCCI are:

- Flexibility: enabling a dynamic, adaptable model,
- Simplicity: do not mandate a large number of requirements for compliance with the specification. Look to provide the lowest common denominator in terms of features and then allow providers supply their own differentiating features that are discoverable and compliant with the OCCI core model,
- Extensibility: enable providers to specify and expose their own service features that are discoverable and commonly understood (via core model).

The specification itself currently comprises of 3 modular parts:

- Core [OCCI_CORE]: This specifies the basic types and presents them through a meta-model. It is this specification that dictates the common functionality and behaviour that all specialisations of it must respect. It specifies how extensions may be defined.
- Infrastructure [OCCI_INFRA]: This specification is an extension of Core (provides a good example of how other parties can create extensions). It defines the types necessary to provide the a basic infrastructure as a service offering.

- HTTP Rendering [OCCI_HTTP]: this document specifies how the OCCI model is communicated both semantically and syntactically using the RESTful architectural-style.

From an architectural point of view OCCI sits on the boundary of a service provider (figure below). It does not seek to replace the proprietary protocols/APIs that a service provider may have as legacy.

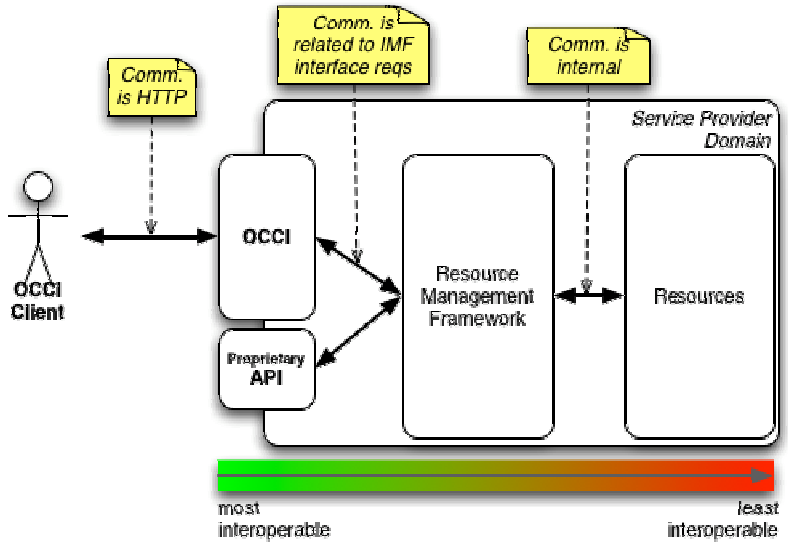


Figure 4 OCCI interface

The main capabilities of OCCI are:

- Definitions (attributes, actions, relationships) of basic types:
  - Compute: defines an entity that processes data, typically implemented as a virtual machine.
  - Storage: defines an entity that stores information and data, typically block-level devices, implemented with technologies like iSCSI and AoE.
  - Network: defines both client (network interface) and service (L2/L3 switch) networking entities, typically implemented with software defined networking frameworks.
- Discovery system - Types and their instances' URL schema (provider can dictate their own) is discovered. Extensions are also discoverable through this system.

- Extension Mechanism allows service providers expose their differentiating features. Those features are comprehended by clients through the discovery system.
- Resource (REST) handling (CRUD) of individual and groups of resource instances
- Tagging & Grouping of Resources
- Dynamic Composition that allows for the runtime addition of new attributes and functional capabilities
- Template support for both operating systems and resource types
- Independent of provisioning system

The current release of the Open Cloud Computing Interface is suitable to serve other models in addition to IaaS, including e.g. PaaS [OCCI_GCDM]. It has wide open source software adoption with many implementations[2] and a number of supporting tools[3]. It is has been recommended by the UK G-Cloud initiative[4], is currently in the process of consideration by NIST[5] in the US and also supported by the SIENA[6] and EGI[7] initiatives here in the European Union. OCCI has also been contributed to significantly by EU FP7 projects including RESERVOIR and SLA@SOI. Forth-coming extension to the specification include those that expose monitoring[8] and SLA capabilities[9].

Resource Allocation SLOs

SLOs are technical clauses in legally bounding documents called Service Level Agreements (SLAs), specifying the terms and conditions of service provisioning. SLOs specify non-functional guarantees of the cloud provider with respect to the virtualized workloads and resources, which the cloud provider offers to its users.

Enterprise grade SLO compliance is one of the critical features that are insufficiently addressed by the current cloud offerings [Reservoir-Computer2011, Reservoir-Architecture2009, SLA@SOI-Architecture10].

In this GE, we focus on **resource allocation SLOs** (RA-SLOs), which guarantee the actual resource allocation according to the nominal capacity of VM instances. They are useful both in the context of elastic and non-elastic services. In particular, RA-SLO specifies that a VM is guaranteed to receive all its resources according to the specification of the VM's instance type with probability p throughout a single billing period. It should be noted that today this mechanism does not exist as part of public cloud offerings.

---

[2] http://www.occi-wg.org/community/implementations

[3] http://www.occi-wg.org/community/tools

[4] http://occi-wg.org/2011/02/21/occi-and-the-uk-government-cloud/

[5] http://www.nist.gov/itl/cloud/sajacc.cfm

[6] http://www.sienainitiative.eu/

[7] http://www.egi.eu

[8] http://www.iolanes.eu

[9] http://en.wikipedia.org/wiki/D-Grid

RA-SLO guarantees are orthogonal to those of the up-time SLOs. Thus, if an up-time SLO guarantees 95.5% of compliance within a single billing period, then RA-SLO guarantees that throughout this time resource allocation is in accordance to the nominal capacity specification with the percentile of compliance p_1 that may be either greater, less or equal to the up-time percentile of compliance p.

Providing RA-SLO guarantees is especially challenging in conjunction with the natural desire of the FI-WARE Cloud Instance Provider to minimize the hardware cost by over-booking resources. Such an over-booking is typically done by multiplexing existing physical capacity among multiple workloads, and relying on the assumption that different workloads would typically achieve peak capacity demand at different times. To guarantee RA-SLO compliance, the IaaS DataCenter Resource Management GE will execute an admission control mechanism that allows validating feasibility of RA-SLO guarantees over time for a given workload before actually committing to these guarantees under the over-booking model.

While this GE will be responsible for enforcement of RA-SLOs, they will be submitted to the IaaS DataCenter Resource Management GE by the IaaS Service Management GE, described in Section 3.2.2.

3.2.1.3 *Critical product attributes*

- Infrastructure scalability
- Resiliency of the management stack
- Enforcement Resource Allocation SLOs (RA-SLOs)
- Optimized placement of virtual resources on physical nodes ensuring high resource utilization

3.2.2 IaaS Service Management

3.2.2.1 *Target Usage*

The IaaS Service Management GE introduces a layer on top of IaaS Resource Manager GEs (both DataCenter and Cloud-edge) in order to provide a higher-level of abstraction to Application/Service providers. Thus, the Service Provider does not have to manage the individual placement of virtual machines, storage and networks on physical resources but deal with the definition of the virtual resources it needs to run an application/service, how these virtual resources relate each other and the elasticity rules that should govern the dynamic deployment or deactivation of virtual resources as well as the dynamic assigned of values to resource parameters linked to virtual resources (CPUs, memory and local storage capacity on VMs, capacity on virtual storage and bandwith and QoS on virtual networks, for example).

3.2.2.2 *GE description*

Overview

The IaaS Service Management GE relies on the concept of vApp and Virtual Data Centers (VDC). A vApp comprises:

- a set of VMs
- optionally, a set of nested vApps

A Virtual Data Center (VDC) maps to a virtual infrastructure defined by the user to host the execution of application/services. A VDC is made up of a number of vApps, and a set of virtual networks and virtual

storage systems. In order to deploy VMs linked to a vApp, we have to provide information about their configuration, needed files to run it, license information, security issues, and so on. In addition to it, the VM can require the specific technology stack installed on top of the Operating System (e.g., DB, Application Server). Finally, the provider can also specify some hardware characteristics (number of CPU, RAM size and characteristics of network interfaces) as well as local storage capacity..

This GE would be the one responsible of the following functions:

- Deployment Lifecycle Management: it coordinates the workflow that is responsible for the deployment and re-deployment (i.e., on the fly change of the service definition not just VMs) of VDCs and applications/services based on an IaaS Service Manifest. It also maintains the service configuration status at runtime, allowing the application of different optimization policies for deployment and scalability control.
- Dynamic Scalability: it will automatically control the elasticity of services by scaling up/down the service in a vertical (adding/removing horsepower such us CPU, memory or bandwidth) or horizontal (adding/removing service nodes replicas) way. Service Scalability would be based on Elasticity Rules defined by the user through the Self-Service Portal/Backend or generated by the Advanced IaaS/PaaS Management GEs (which in turn are configured based on input provided by the user through the Self-service Portal/Backend). To perform this function, this GE will rely on monitoring and accounting data provided by the Monitoring/Accounting GE.
- Federation and Interoperability: In a private Cloud schema there could be some clusters providing different virtualization technologies or different service quality levels (from best effort to high availability). In a hybrid Cloud schema, local infrastructures are federated to remote (public or private) Clouds to cope with peaks of demand which cannot be satisfied locally. In both cases, this layer will allow to distribute service components among different local virtual Hosting Centers or among local and remote virtual Hosting Centers using functions offered by the IaaS DataCenter Basic Resource Management GE.

The IaaS Service Management GE will offer as "north" interface a standard Cloud Service Management API which will enable to program the creation of the IaaS Service Manifest definitions based on DMTF's OVF (Open Virtualization Format) and gathering of monitoring data from GEs in lower layers of the Cloud Hosting Reference Architecture (IaaS DataCenter and Cloud-edge Resource Management GEs). Usage of this API, together with definition of portable VM images would guarantee portability and interoperability of deployments. That’s why one of the goals related to the IaaS Service Management GE in FI-WARE will have to do with pushing standardization of this API within DMTF’s Cloud Management Working Group (CMWG).

IaaS Service Manifest

The input to the IaaS Service Manager GE is the service manifest, where the application/service provider specifies the features and requirements of the virtual infrastructure hosting the execution of a number of applications/services. The service definition is an XML document containing information such as:

- Service features (application and features, software properties, etc.).
- vApps, virtual networks, virtual storage systems and other virtual resources required to deploy the service
- Hardware requirements to deploy each VM
- Restrictions and service KPIs for scaling up or down

As FI-WARE works with different sites and with a variety of services, it is required that virtual machines will be portable between sites. It is required to guarantee interoperability in service and virtual machine definitions between the sites. In a cloud environment, one key element of the interaction between Service Providers (SPs) and the infrastructure is the service definition mechanism. In IaaS clouds this is commonly specified by packaging the software stack (operating systems, middleware and service components) into one or more virtual machines hosting an application/service, but one problem commonly mentioned is that each cloud infrastructure provider has its own proprietary mechanism for service definition. This complicates interoperability between clouds and locks a service provider to a particular vendor. Therefore,

there is a need for standardizing the service definition in order to avoid vendor lock-in and facilitate interoperability among IaaS clouds.

The Open Virtualization Format (OVF) [OVF 08] is a standard from the DMTF [ServiceManifest 10] which can provide this service manifest. Its objective is to specify a portable packaging mechanism to foster the adoption of Virtual Appliances (vApp) as a new software deploy and management model (e.g. through the development of virtual appliance lifecycle management tools) in a vendor and platform neutral way (i.e., not oriented to any virtual machine technology in particular). OVF is optimized for distribution and automation, enabling streamlined installations of vApps.

Deployment Lifecycle Management

As previously mentioned, the IaaS Service Management GE coordinates the deployment and redeployment (on the fly change of the service definition) workflow and maintains the service configuration status at runtime, allowing the application of different optimization policies for deployment and scalability control.

As commented above, FI-WARE considers the concept of VDC as a set of virtual machine, network and storage support as a whole. The Service manager is in charge of managing the VDC as a whole instead of independent virtual machines and networks. The IaaS Service Manager GE is able to process the IaaS Service Manifest it receives as input and translate it into the right requests to the IaaS DataCenter Resource Management GE, distributed Cloud-edge Resource Managers and even third IaaS Cloud providers. These requests cover the deployment of the different elements that comprise a VDC, as well, as other operations (undeployment and so on). Finally, the IaaS Service Manager GE is in charge of managing the service lifecycle as presented in next figure.

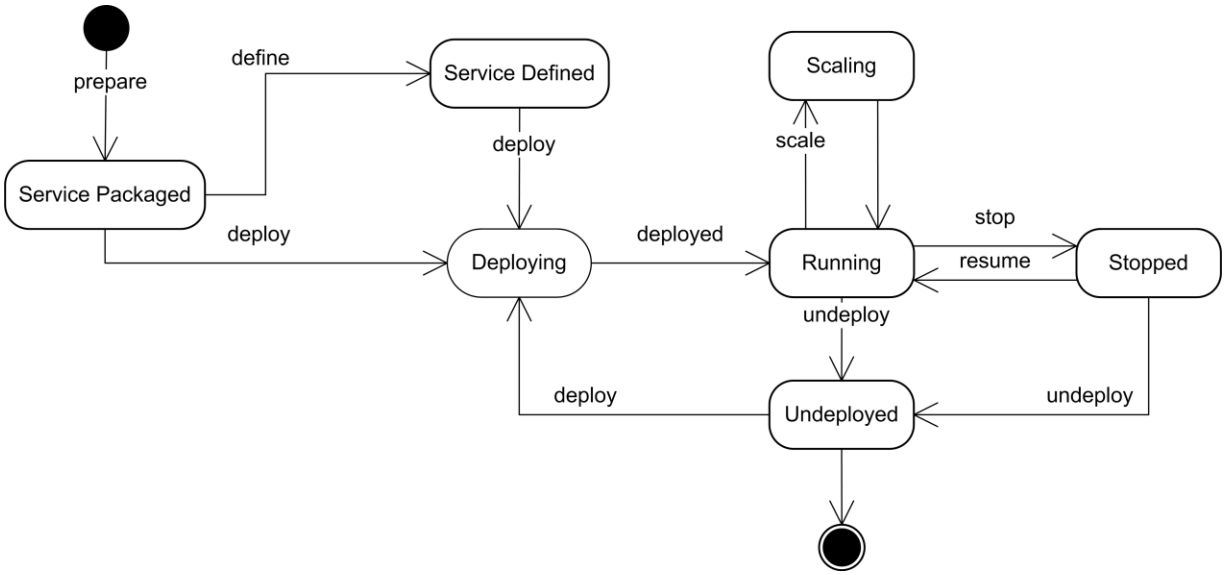


Figure 5 VDC Service lifecycle

The typical lifecycle of a VDC service in the Cloud is shown in Figure 5It can be observed how one needs to develop and package appropriately a service for the Cloud [Vaquero et al. 2011], i.e., the transition from our in-house premises to the Cloud is no yet straightforward. After the service is packaged, the next step takes us to the definition of what we want our service to be. In IaaS Clouds, this second step can be skipped as they offer a virtual machine level API that constrains the specification of an application (understood as a set of services or virtual appliances) to a set of detached “commands” for every VM. The next step consists on deploying the service based on requests upon the IaaS DataCenter Resource Manager GE, distributed Cloud-edge Resource Management GEs and third party IaaS Clouds. Having the service deployed leads us to the running phase, the one in which the application will be laying most of the time and that in which and appropriate control on the services' behaviour is most desirable. When the service is running, two

prominent processes are identified: 1) monitoring the services status in order to 2) react, by scaling up/down [Cáceres et al. 2010], and keep the promised quality of service and economic performance. At a later stage, the service can be stopped (to be again resumed) or undeployed and destroyed.

Dynamic Scalability

The cloud computing automated provisioning mechanisms can help applications to scale up and down systems in a way that performance and economical concerns are balanced. Scalability can be defined as "the ability of a particular system to fit a problem as the scope of that problem increases (number of elements or objects, growing volumes of work and/or being susceptible to enlargement)" [Cáceres et al. 10]. Under the FI-WARE context, scalability is managed at the service level, i.e., by the IaaS Service Manager GE. The actions to scale may be classified in [Cáceres et al. 10]:

- Vertical scaling by adding more horsepower (more processors, memory, bandwidth, etc.) to deployed virtual resources. This is the way applications are deployed on large shared-memory servers.
- Horizontal scaling by adding more virtual resources. For example, in a typical two-layer service, more front-end VMs are added (or released) when the number of users and workload increases (decreases).

The service manager, besides managing the service, has to manage monitoring events and scalability rules. It is responsible for avoiding over/under provisioning and over-costs. The IaaS Service Manager GE protects SLOs specified for services using business rules protection techniques. It provides a means for users to specify their application behaviour in terms of vertically or horizontally scaling virtual resources [Cáceres et al. 10] by means of elasticity rules [Chapman 10]. The elasticity rules follow the Event-Condition-Action approach, where automated actions to resize a specific service component (e.g. increase/decrease allocated memory) or the deployment/undeployment of specific service instances are triggered when certain conditions relating to these monitoring events (KPIs) hold. In certain circumstances, some Resource Allocation - SLOs are passed to the IaaS DataCenter Resource Manager so that it can govern placement decisions.

Regarding the scalability, our solutions is much more flexible, richer and the actions are not only limited to up and down scaling [Vaquero et al. 2011] compared to other solutions like Cloud Formation, Right Scale or Amazon Elastic Beanstalk. Other differentiation aspect from this solution is that it is based on standard solutions and does not use proprietary APIs.

Service Monitoring

Every system needs to incorporate monitoring mechanisms in order be able to constantly check the performance of the system. This is especially true in service clouds that are governed by SLOs, which are measurable technical artefacts derived from the clauses of Service Levels Agreements (SLAs) and so the system needs to be able to constantly check that the performance adheres to the terms contracted. In order to guarantee the SLOs of the SLAs, the service manager offers scalability mechanism to satisfy the customer's demand. This scalability is driven by service monitoring metrics. The service monitoring metrics can vary from virtual hardware attributes, to Key Performance Indicators (KPI), event to platform-level metrics installed in the virtual machine and also at the service KPI level (this is the case for PaaS platforms)

Federation and Interoperability

In a private Cloud schema there could be some clusters providing different virtualization technologies or different service quality levels. Also, in hybrid Cloud schema, local infrastructures are federated to remote (public or private) Clouds to cope with peaks of demand, which cannot be satisfied locally. In both cases, a service or a virtual machine can be deployed in different Cloud provider in a federation scenario. However, the Cloud providers use proprietary interfaces and data models which introduce problems related to heterogeneity in operations and data. Thus, there is a need for interoperability in order to avoid ad-hoc developments and the increase of the time-to-market. The solution should offer a common access to service

providers based on the existence of a common API, since it allows to aggregate different Cloud providers' APIs. In order to guarantee interoperability, the data model in the aggregator API should be standard.

Cloud Service Management API

The Cloud Service Management API allows to deploy service manifests, or fragments of service manifests, based on the standard Open Virtualization Format (OVF)defined by the DMTF [OVF 08]. This API is a RESTful, resource-oriented API accessed via HTTP, which uses XML-based representations for information interchange. It will be based on the TCloud API [TCloud 10] being proposed at the CMWG of the DMTF, which is based on already consolidated OVF specifications and the vCloud specification [VMWare 09] published by VMware and submitted to the DMTF for consideration. It is the goal that the Cloud Management API evolves to align with specifications approved at the CMWG in DMTF.

The Cloud Service Management API will define a set of operations to perform actions over:

- Virtual Appliances (VApp), which is a Virtual Machine running on top of a hypervisor,
- Hw resources the virtual hardware resources that the VApp contains,
- Network both public and private networks, and
- Virtual Data Center (VDC) as a set of virtual resources (e.g. networks, computing capacities) which incarnate VApps, which are owned by Organizations (Org) (independent unit).

The Cloud Service Management API will define operations to perform actions over above resources categorized as follows: Self-Provisioning operations to instantiating VApps and VDC resources and Self-Management to manage the instantiated VApps (power on a VApp…). In addition, it provides extensions on monitoring, storage, and so on.

In addition, the Cloud Service Management API will be focused on adding network intelligence, reliability and security features to cloud computing empowered by enhanced telecom network integration [TCloud 10]. Moreover, it will aim to extend current cloud computing models providing more flexibility and control to cloud computing customers. DMTF defines cloud computing as "an approach to delivering IT services that promises to be highly agile and lower costs for consumers, especially up-front costs". This approach impacts not only the way computing is used but also the technology and processes that are used to construct and manage IT within enterprises and service providers.

Compatibility for the main operations and data types defined in vCloud [VMWare 09] will be maintained in the Cloud Service Management API, but it will provide extensions for advanced Cloud Computing management capabilities including additional shared storage for service data, network element provisioning (different flavors of load balancers and firewalls), monitoring, snapshot management, etc. The following table summarizes main characteristics currently supported in the TCloud API and therefore going to be supported by the Cloud Service Management API

Cloud Service Management API characteristics

Characteristics	Description
Aggregation paradigm	The auto-browser API shows the extension and functionalities provided by the implementation. If there is some support with missing operations, the interface will be implemented and OperationNotSupportedException will be thrown when the client calls non-supported operations.)
API type	The API will be a RESTful API. A Java binding easing programming to this programming will also be provided. Binding to other languages may also be considered.
Support for 3rd party	The API implementation will include drivers to handling access to the IaaS

Characteristics	Description
driver integration	DataCenter Resource Manager GE and 3rd party GEs exporting OCCI. However, additional drivers (to handle Amazon EC2, for instance) will be easy to add.. There is the intention to support drivers for Amanzon EC2, OpenNebular, Emotive and Flexiscale.
VM type	The data model is based on OVF. It will be possible to define: • Virtualization technology (xen, kvm...) • The URL where the image is located • Virtual hardware parameters (CPU, ram, disk) • Network information (public, private) • Software configuration information (contextualization information) • The definition of the service as a whole (not only VMs) • Scalability information • Storage information (NAS, SAN...)
VM image	No restriction about supported OS. It is possible to deploy a VM or attached a image to be deployed.
VM lifecycle	Supported states: deploy, start/stop, reboot, suspend, activate, undeploy Reconfiguration support (state constraints): to stop a VM and deploy a new one.
Storage	Support for various storage types: NAS, SAN… including integration with CDMI API. Lifecycle, operations: • create/delete • attach/detach • snapshots
Network	Supports declaration and configuration of resources linked to Public and Private Virtual Networks.
User & account management	The customer concept will be related to the VDC concept. It is possible to add/delete/modify... a VDC
Advanced features	Monitoring Multi-VM deployment (like OVF)
Local resource accounting	No. It has not information about resources (it is scope of the Cloud Providers)
Support for lengthy operations	The API will support asynchronous interactions providing a task Id. Polling and pushing mechanisms are also provided

3.2.2.3 *Critical product attributes*

- Management of the service as a whole (considering virtual machines, networks and storage support).
- Management of the overall service lifecycle.
- Scalability at the service level with powerful elasticity rules.
- Service monitoring not only at Infrastructure level but Service and Software (installed in the VM) KPI levels.
- Interaction with different IaaS providers according to richer placement policies.

3.2.3 PaaS Management

3.2.3.1 *Target usage*

The PaaS Management GE will provide to the users the facility to manage their applications without worrying about the underlying infrastructure of virtual resources (VMs, virtual networks and virtual storage) required for the execution of the application components. This means that the user will only describe the structure of the application, the links among Application Components (ACs) and the requirements for each Application Component. This GE will deal with the deployment of applications based on an abstract Application Description (AD) that will specify the software stack required for the application components to run. This software stack will be structured into Platform Container Nodes, each linked to a concrete software stack supporting the execution of a number of Application Components and allowing to structure the Application into several Software Tiers. Besides, the AD will describe the Elasticity Rules and configuration parameters that may help to define how initial deployment is configured (generation of Service Manifest definition) and what initial elasticity rules will be established.

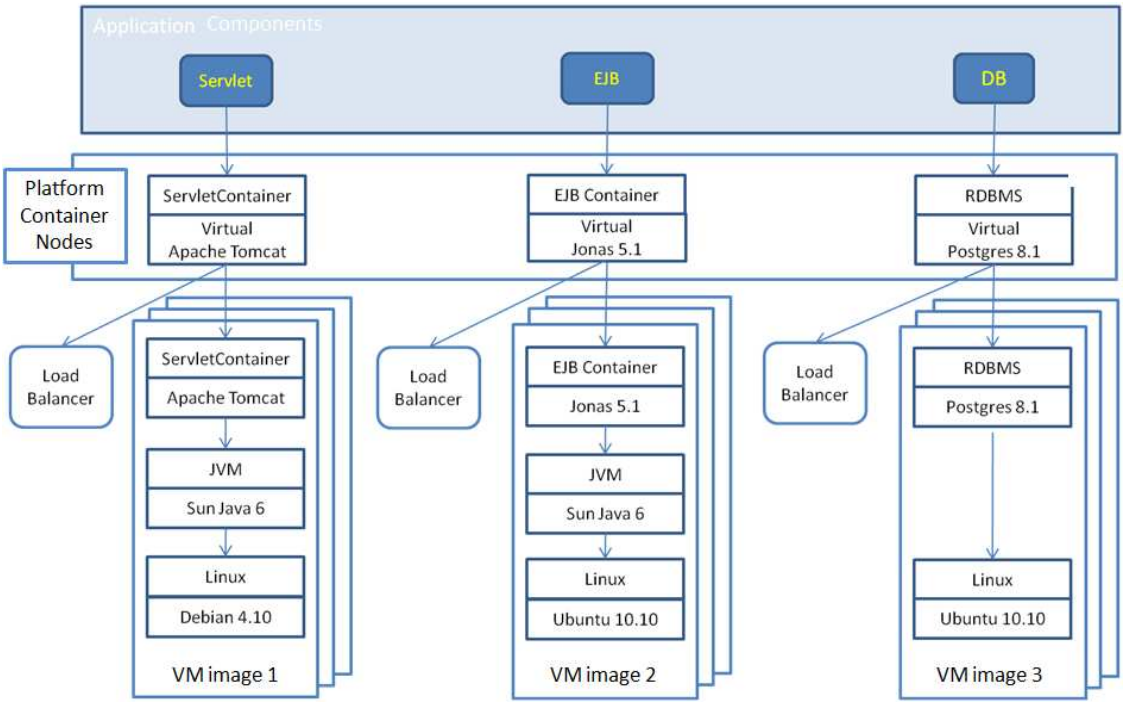


Figure 6 Abstract Application Description and their mapping into deployed VM images

The main difference between IaaS and PaaS is that IaaS manages the structure and lifecycle of the virtual infrastructure required for applications to run. PaaS on the other hand, manages the structure and lifecycle of the applications and platform containers..

3.2.3.2 *GE Description*

Overview

Application description

In the context of a PaaS provider, a description of the Application is required in order to deploy the application onto the platform. This Application Description (AD) will contain:

- The structure of the application: The Application is structured into Application Components, which in turn are distributed across an number of connected Platform Container Nodes.
- Requirements in terms of:
  - Software stack linked to each Platform Container Node: webserver technology, application server technology, services composition technology, database technology, etc. The catalogue of available technologies and products will be provided and managed by the platform.
  - Resource execution requirements in terms of CPU speed, disk storage, storage types linked to each Platform Container Node
  - Network resource requirements. This level only specifies these requirements and furthermore the IaaS Service Manager GE will be responsible to manage them.
  - Connection with FI-WARE GEs. In case the FI-WARE Cloud Instance Provider offers additional GEs to be used by the application, like the Data/Context Management GEs or Security GE. In this case, the description of the application should contain details about usage of services provided by the GEs that allow the platform to configure the proper access of the application to these services.
  - Elasticity rules. The client can specify the way the application scale based on measurements.
  - Other requirements can be also included depending on the capabilities of the platform, like backup policies, or placement policies, etc.

The description of the application at this stage is abstract in the sense that the user describes the application without considering how such description will map into a final IaaS Service manifest or changes into existing IaaS deployments. Thus, the user will not describe how Platform Container Nodes will map into VMs.

The users will also define the Elasticity Rules based on KPIs that are relevant from an application perspective (i.e. maximum number of DB connection, maximum number of concurrent access to an Web Server, and so on). It is the PaaS Management GE that knows the way in which these application-oriented elasticity rules will be mapped into elasticity rules handled by the IaaS Service Manager GE.

One possible standard to use as a basis for these abstract ADs is the OVF (defined by DMTF) [OVF 08], which is also used for the description of VDCs in the IaaS context. The support of PaaS concepts could imply the future extension of the OVF to include concepts relevant to PaaS as described above. FI-WARE will carefully monitor results relevant to this matter coming out from the 4CaaSt EU FP7 project.

Deployment Runtime design

Before the application is deployed, a more detailed descriptor of the application is designed based on the requirements specified by the client.

The input of this process will be the Application Description (AD). Based on the requirements imposed on the different Application Components (ACs) the PaaS Manager GE will generate an Application Deployment Descriptor (ADD) which will comprise an initial IaaS Service Manifest or set of changes to existing IaaS deployment that will be delivered to the IaaS Service Management GE. It will also comprise information necessary to install, configure and run the different Application Components. Some examples of decisions taken while mapping an AD into information to be submitted to the IaaS Service Management GE could be:

- One Platform Container Node requires a large amount of CPU speed and that cannot be provided by one single VM so the Platform Container Node is mapped into a number of VMs with a load balancer in front of then.
- Two equivalent Platform Containers Nodes (defined over a compatible technology stack) have very small requirements so the system decides to map both into a single VM.

Note that these decisions affect the final network configuration and also the definition of software products included into the different VMs on which the Application Components will finally run.

The process described above is illustrated in the following figure. Once again, the Application Deployment Descriptor (ADD) may be expressed based on an extension of OVF.

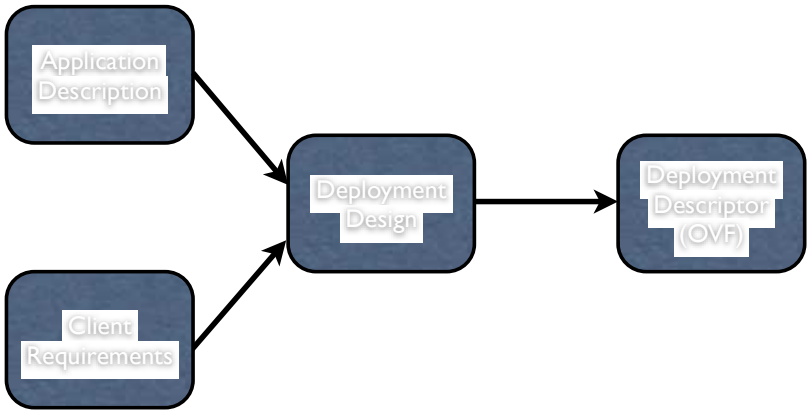


Figure 7 Deployment design process

Runtime and Application deployment

Once the Deployment Runtime design is done and the final Application Deployment Descriptor (ADD) generated, the actual provisioning and deployment of the application is performed. This implies planning the sequence of steps that will be required to implement that deployment:

1. Interaction with the IaaS Service Management GE, for setting-up the VDC on top of which Application Components will run. This will require passing a IaaS Service manifest or a set of changes to existing IaaS deployment to the IaaS Service Management GE.
2. Installation of the software linked to Application Components and Provision of the connection to other FI-WARE GE services (e.g., Data/Context Management GEs) that the application is going to use.
3. Start of the different Application Components on which the Application is structured.

Once the first step is finished, the VMs, virtual networks and virtual storage capabilities needed by the Application will be deployed into the IaaS. A given Platform Container Node will be mapped into a single or several VMs that will be instances of VM images taken from the VM image repository. Therefore, the software stack linked to the Platform Container Node is mapped into software configured in the images of the corresponding VMs.

After the VDC is set up, some parameters linked to VMs will be configured for the correct installation of Application Components or the proper connection to complementary FI-WARE GEs (e.g., Data/Context

Management GEs). Besides, the installation of the Application Components themselves will be carried out. This process of setting configuration parameters and installing additional software on an existing VM is called contextualisation.

Once the Application Components are installed and correctly configured, the application will be started. The order in which the different application components are started is defined in the application description (AD). This will allow to handle some dependencies between Application Components.

During the execution of the application, monitoring information will be collected and used for various reasons: like scaling of the application, SLO Management, etc.

The application will be finally terminated by the client (final user), this process will free the resources assigned to the application (inside the IaaS and the cloud services used).

Adaptation and scalability of deployed application/services

The PaaS Management GE will allow applications to adapt during execution to the changing demands of users or resource shortages. This could be linked with the SLOs of the application's SLA in place or scalability rules defined by the application provider.

The scalability capability is closely related with the monitoring system to collect and process the different KPIs that could affect the scaling of the application.

The scalability can affect application components or platform elements (products implementing the software stack linked to Platform Container Nodes) or complementary FI-WAGE GE services integrated as Cloud Services. This layer should know about the characteristics of the different elements to know how they scale and their limitations.

Not only the scale-up (in the same physical host) or scale-out (to multiple VMs) will be supported, also the shrinking of resources will be performed when the environment of the application allows this.

There will be an interaction between the scalability components, the provisioning and deployment layer to create, stop, destroy, and reconfigure VMs, infrastructure and/or network resources.

The PaaS Management GE will drive resource allocation using the underlying IaaS layer according to the elasticity rules.

3.2.3.3 *Critical product attributes*

- A common specification for the definition of the complete application structure and its requirements.
- The ability to automatically design the final structure of the deployment based on the abstract description of the application and its requirements and restrictions.
- Automatic management of the low level concepts, allowing the client to focus on the application. The deployment of application containers, database, load balancers and the scalability of the application will be managed by the PaaS layer.
- Common interface API for multiple and different IaaS providers.

3.2.4 Object Storage

3.2.4.1 *Target usage*

The Storage Hosting GE comprises a storage service that operates at a more abstract level than that of block-based storage (e.g. iSCSI-, AoE-type technologies). Rather than storing a sequence of bits and bytes, it stores items as units of both opaque data and meta-data. An example of object-based storage is Amazon's S3 service offering where data (objects) are stored in buckets (containers of objects).

The users of such a service will be both the FI-WARE Cloud Instance Provider (FCIP) and the FI-WARE Cloud Instance User (FCIU).

- FCIP usage: The CHP can take two roles: one as consumer of the service and another as its manager. From a consumer perspective, the FCIP will use this system in order to store certain types of data. A good example of this is in the storage of monitoring, reporting and auditing data. That data could then be made available to the FCIUs or not depending on the wants of the FCIP. The Object storage service could also be used as a virtual machine staging area. This has two aspects, the FCIP and FCIU aspects. In the case of the FCIP, a FCIU will upload a virtual machine image to the object storage service and once received, the FCIP will make this virtual machine image available to the FCIU in order to satisfy a particular customized virtual machine request (the case here is that the virtual machine images that the FCIP offers are not sufficient and the user wishes to supply their own). From a management perspective, the FCIP will expect that the system will require as little maintenance as is possible. This entails that where:
  - Stale data exists it should be purged,
  - Deactivated accounts present they are removed,
  - Corrupt data is discovered, it is replaced with a valid replica
  - Issues are discovered, they are raised to an automated service that will attempt to resolve. If they cannot then notifications to the FCIP should be sent.
  - Necessary a full statistics system should be available to inspect system and the user's utilisation of the system
  - New hardware (storage capability) is required that it can be easily added to the system without any drop in service. This will allow the storage capacity to grow over time.
- FCIU usage: The FCIU will use the object storage service as a means to distribute static content rather than incur the additional load of serving static content from an application. Taking this approach allows the FCIP to optimize the distribution of those files. The FCIP can also use this as a building block to offer further content distribution network capabilities. The FCIU could also use the object storage service as a means to supply a customized virtual machine that only they have access to (the storage is isolated by user). This would follow in a similar fashion to how customized virtual machine images are supplied on Amazon EC2.

3.2.4.2 *GE description*

The Object Storage GE in FI-WARE will be composed of loosely coupled components. A potential architecting of this would be to separate the components for API handling, authentication, storage management and storage handling. Separating the main functional components of the system will allow for scaling. Given that the demand on storage systems is to ever increase the storage capacity, the system should scale with commodity hardware horizontally. A high level view of the Architecture of the Object Storage GE could look like:

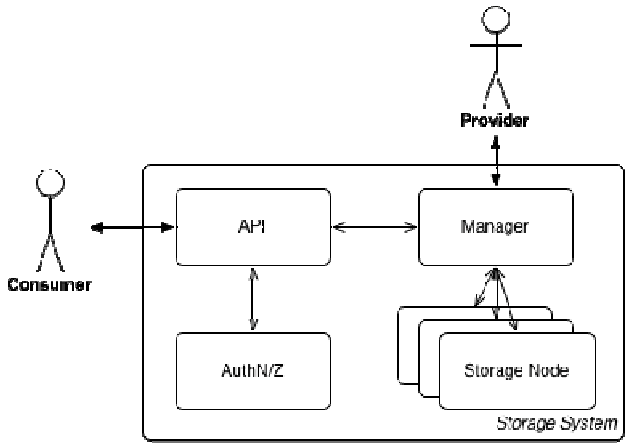


Figure 8 Object Storage GE

The OS GE should support integration of external AuthZ (authorization) and AuthN (authentication) so that integration with Security GEs can easily be accomplished.

In order to remain interoperable and mitigate lock-in, the Object Storage GE will rely on open, patent unencumbered standards for the definition of interfaces and where appropriate will seek to integrate with other related and complimentary ones. These interfaces should implement techniques to allow the migration of data held under the management of the OSS to other services (see CDMI). This is core to providing this as a generic enabler. Currently the most prolific cloud-oriented storage standard is the SNIA's CDMI specification. This is a specification that is well aligned to other specifications such as DMTF's OVF and OGF's OCCI.

The Object Storage GE will offer the basic interaction of CRUD to the objects it stores and manages. Along with this, the OS GE will:

- Store objects (any opaque data entity) along with user and provider specified metadata into logical (e.g. hierarchical filesystem or tagging) and/or physical groupings (e.g. location). Those grouping in some systems are known otherwise as 'containers'. These stored objects must be then listable according to how they have been grouped.
- Allow users through meta-data, specify particular qualities of the service (e.g. number of replicas, geographic location of the data) that must be adhered to. This should be done on not only a per-object basis but also on a group of objects.
- Enable versioning of objects. Every time there is a modification (update, delete) to an existing object, the previous copy to the now current object should be kept.
- Provide a means to retrieve metrics associated with objects and sets of objects.
- Provide a means to retrieve audit and account information such as access logs and billing information
- As mentioned above, we should consider providing other optimal endpoints for integration with other provider infrastructural services
- Offer a discovery mechanism so that potential clients can discover what service offerings are available. It is taken for granted that the service will be a storage service however a client will need to know what tuneable parameters and service specialisations are available. This is an important interoperability feature.

From a management perspective, the Object Storage GE will:

- Provide system-wide policies. Though a user might specify certain quality of service attributes through an object's meta-data, it might violate a system-wide, provider set policy. The reconciliation of user and provider requirements should be considered and processed by the system. This aspect would include the notions such as provider-imposed limitations such as per-user rate limiting.

- Allow for the use of commodity hardware and for new hardware to be easily added to the system at runtime to deal with a growing system. This growth of the system entails that it should be architected so that is scales horizontally.
- Integrate with different back-end storage systems
- Integrate with different authentication and authorisation systems. The system should have a pluggable mechanism to enable this.

CDMI API

The SNIA Cloud Data Management Interface (CDMI) [CDMI 11] is the interface that it is used to create, retrieve, update, and delete (CRUD) data contents from the cloud. The CDMI standard uses RESTful principles in the interface design. As part of this interface, the client will be able to discover the cloud storage capabilities offering and to use this interface to manage containers. These containers have the storage units that have the data placed in them. Besides it, Containers may set the data system metadata through this interface.

The main characteristic of the CDMI is that it is planned not only to move data but (and most importantly) metadata from cloud to cloud. When managing large amounts of data with differing requirements, metadata is used to reflect those requirements in a way that underlying data services may differentiate their treatment of the data to meet those requirements.

Management and administrative applications may also use this interface to deal with containers, domains, security access, and monitoring/billing information. Moreover, the storage functionality is accessible via legacy or proprietary protocols too in order to allow the utilization of legacy cloud storage system.

This standard manages important attributes from the point of view of cloud services, like pay as you go, the illusion of infinity capacity (through elasticity), and finally use and management simplicity which allow to

The common operations that CDMI manage are the following:

- Discover the Capabilities of a Cloud Storage Provider
- Create a New Container
- Create a Data Object in a Container
- List the Contents of a Container
- Read the Contents of a Data Object
- Read Only the Value of a Data Object
- Delete a Data Object

3.2.4.3 *Critical product attributes*

- Horizontally scaling service that allows for the dynamic addition of commodity hardware
- Can integrate a number of different backend storage facilities
- Implements open and standard API to clients
- Offers efficient management access
- Can be integrated with other provider infrastructural services (TBD)
- Offers storage services with client specified qualities of service
- Enables provider policies to govern client requirements
- Pluggable back-end authentication services
- Monitoring, logging and statistics system for both consumer and provider

3.2.5 IaaS Cloud-edge Resource Management

3.2.5.1 *Target usage*

At first glance computing and storing in the cloud opens unlimited perspectives to the services proposed to the end user: the cloud virtual platform, made of a mass of high-performance servers connected via many high speed links, seems to be an inexhaustible resource in term of computing and storage capabilities.

However, the link between the cloud and the end consumers (as well as many SMEs) appears to be a weak point of the system: it sometimes may be unique (therefore a single point of failure) and it may offer a relatively low bandwidth in some scenarios. Actually, typical ADSL bandwidths are in the range of several Mbit/s while a private LAN like a home network at least offers 100 Mb/s and more and more commonly 1 GB/s. One could indeed argue that new home connection based on optical fibre technologies shall increase this bandwidth to 100 Mb/s and even more. However, we can reasonably consider that, in the constant movement of technology improvement, carrying a bit inside a home network shall remain less costly and more effective than carrying a bit between the home network and the cloud. And, even if the bandwidth goes up, the uniqueness of the link remains.

For these reasons, it may be helpful to use an intermediate entity, which we call “cloud proxy”, located in the home system and equipped with storage and cloud hosting capabilities. The cloud proxy is the cloud agent close to the user. It may host part of a cloud application, therefore handling communications between cloud applications an the end user, even when the communication with centralized DataCenters is down or the user is not active anymore; it may provide intermediate storage facilities between the user and the cloud, etc.

The cloud proxy may first be a hardware device with two advantageous characteristics. First, it is always powered on making possible a permanent connection with the cloud, independently of the presence of the end-user. Secondly, it is connected with each device of the home network making possible that each end user, whatever his device is, can take benefit of the cloud proxy presence. Such hardware with these two characteristics typically corresponds to the home gateway. The home gateway (i.e. the cloud proxy) may be owned by the ISP (typically playing the role of FI-WARE Cloud Instance Provider or privileged partner of the FI-WARE Cloud Instance Provider) which makes it available for the user.

The cloud proxy may also comprise a number of software components. Connecting each device in an appropriate manner (i.e., with the ad hoc protocols) may require computer skills if this operation is not automated. The software part of the cloud proxy would be owned by the FI-WARE Cloud Instance Provider (who might be the ISP or a privileged partner of the ISP, therefore guaranteeing a trusted environment execution). One purpose of the software “cloud proxy” may be to ensure the automation of the connection to the diverse devices making the home equipment. It may also to provide the appropriate computing platform to host a program that the cloud may advantageously chose to download and execute locally on the cloud proxy. This computing platform may include middleware through which local applications and cloud applications can interact in the purpose to offer the best service to the end-user.

The picture below illustrates the here outlined concept of cloud proxy.

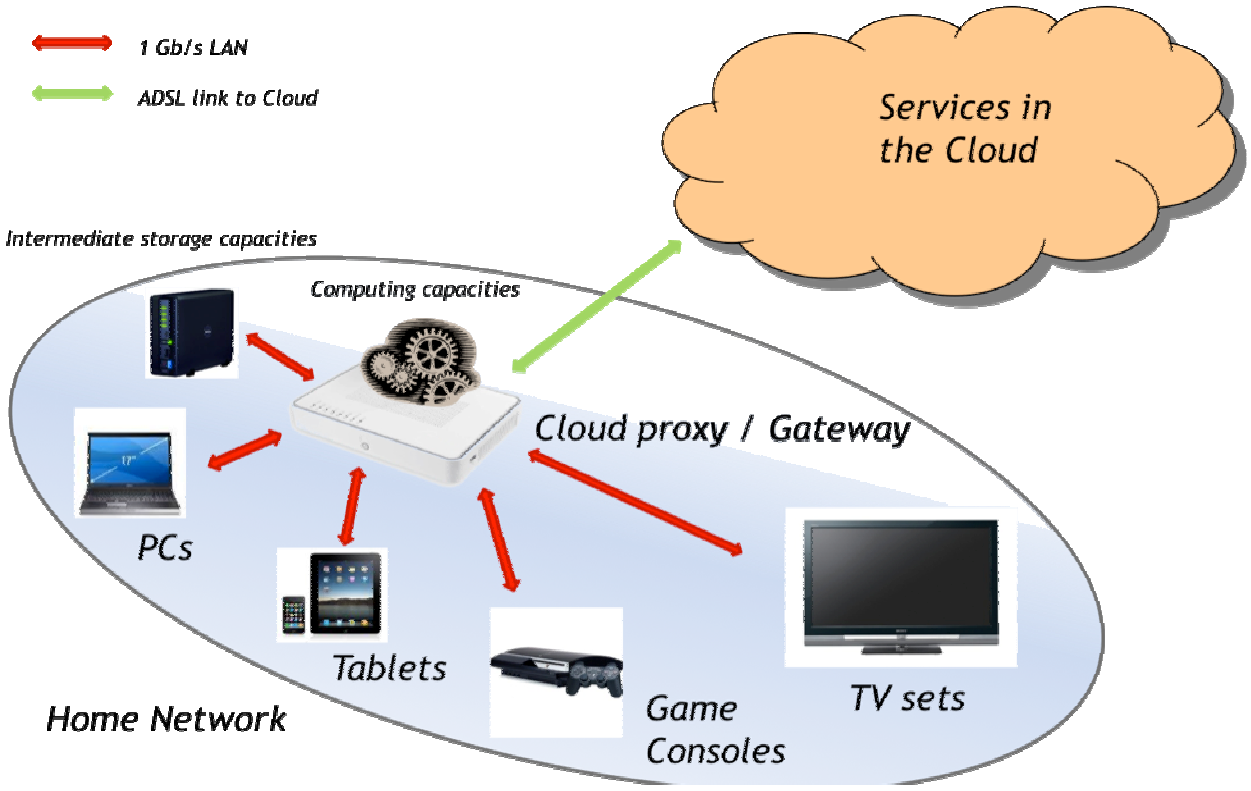


Figure 9 Cloud Edge and Cloud Proxy

Application developers shall be aware of the existence of a cloud proxy, as far as it may help them to build applications/services providing a better experience to the end-users. Cloud proxy functions in FI-WARE will be available to Application developers or other GEs in FI-WARE through a set of APIs that still have to be defined. The IaaS Cloud-edge Resource Management GE comprise those functions that enable to allocate virtual computing, storage and communication resources in cloud proxies. It provides an API that may be exposed as part of the Self-Service API provided by FI-WARE Cloud Instances. Given the fact that it will also allow to deploy Virtual Machines (VMs) that comprise a standard technology stack, they shall also be considered as one element of the PaaS offered to application/service developers.

Several use cases are given here to illustrate the concept of cloud proxy presented in the here above chapter and how applications developers may take profit of it.

From Cloud to User: download a Video On-Demand (VOD) catalogue

The user has subscribed a VOD service. In addition to the simple delivery of movies, the VOD service also includes a browsing service including the visioning of trailers. The reaction speed of the browsing service is a key point of its attractiveness. Storing locally the VOD database, in particular trailers (at least part of them), helps to improve the reaction speed, because access to the database is through the high speed private LAN instead of the ADSL link. The VOD cloud application may download on the cloud proxy a part of its database and an associated application which offers to the user a browsing service. The exact inter-operation way between the VOD cloud application and the downloaded browsing application has to be specified.

The VOD application developer shall be aware that the cloud proxy has storage capabilities and is able to store part of the VOD database. Ways for getting or negotiating how many storage capabilities the cloud proxy offers to the VOD application have to be specified. Knowing the amount of storage available on the cloud proxy, the VOD application may decide which part of its data shall be downloaded and where (on the cloud proxy). When the user shall request for a piece of data downloaded on its cloud proxy, the

application shall re-address the request to the corresponding local storage. Exact mechanism for re-addressing has still to be specified.

From User to Cloud: upload personal pictures

The user has subscribed a picture storing and sharing service (typically like Picasa). He wants to upload a picture album, i.e. a certain number of pictures. Due to the relative low bandwidth offered by the ADSL link, total time for uploading all the pictures in the cloud may represent several hours, constraining the user to remain online several hours. Alternatively, pictures to upload may be temporarily stored on the intermediate storage capabilities offered by the cloud proxy. Time for uploading pictures from the user device to the cloud proxy may be relatively short (some minutes). The cloud proxy will then be in charge of uploading the pictures in the cloud as a background task, letting the user free to leave the home network. Exact organization and relationships between the user device, the cloud proxy and the service in the cloud have to be specified.

The service may be offered to the end-user by cascading two applications: one running on the cloud proxy, which offers an interface to the end-user and ensures the intermediate storage; and the second one running one the cloud which ensures the final backup. The application running on the cloud proxy may be provided by the application/service provider and the combination of the two applications may be seen by the user as one cloud application.

Peer-Assisted application: offload a distributed storage system

Storage services in the cloud may require important storage resources. To avoid a huge increase of storage capabilities need in the cloud, the idea to offload part of the stored data down to end users has been developed. Distributed storage systems (with more or less storage in the cloud) are proposed: the user exchanges part of his storage capabilities with storage capabilities in the distributed storage system. In other words, the distributed storage system organizes the exchange of data between users, a user hosting data belonging to other users while this user's data is also hosted by other users. One important element of such a system, beyond the amount of storage capabilities which a user makes available to the system, is the availability of these storage capabilities. A user only present one hour per day, thus making available his storage capabilities to the system only one hour per day, contributes less than a user present 24 hours a day. However, to be present 24 hours a day is a strong constraint for user. It is not for a cloud proxy which could make available its storage capabilities to the system, on behalf of the user. And, for its part, the cloud proxy could make available to the user the storage space thus gained in the distributed storage system.

3.2.5.2 *GE Description*

The cloud proxy matches to a hardware device where a number of virtual computing, storage and communication resources can be allocated to support the execution of application components or even full applications. The IaaS Cloud-edge Resource Management GE corresponds to software running on the cloud proxy that will enable the deployment and lifecycle management of Virtual Machines comprising application components or full applications plus technology stack (operating system, middleware, common facilities) they require to run.

The IaaS Cloud-edge Resource Management GE comprises a VM management module that is in charge to start, stop and monitor execution of a VM when required. The VM to be deployed will be based on a VM image previously downloaded based on functions also supported by the IaaS Cloud-edge Resource Management GE. Communication with this VM management and VM image download modules may be based on an APIs capable to rely on a optimized use of communications between the cloud proxy and cloud data centers. The exact specification of this API has still to be defined. Using this API, the IaaS Service Management GE should be capable to deploy service manifests (deployment descriptors) which include the distribution of applications partly in centralized data centers and partly on a highly distributed network of cloud proxies.

Note that, to some extend, the IaaS Cloud-edge Resource Management GE software that run on cloud proxies play a role similar to that of the IaaS DataCenter Resource Management GE but being able to work at the scale of a single device (the home gateway) or group of devices (federated home gateways). That's why to some extend a cloud proxy may be referred also as a "nano-datacenter".

FI-WARE will not only define and develop the software linked to the IaaS Cloud-edge Resource Management GE but also software linked to middleware technologies and common facility libraries that will be used in VM images to be deployed in cloud proxies. This way, cloud proxies become one type of Platform Contanier Node in the FI-WARE PaaS offering. Middleware technologies will ease communication between applications running on the cloud proxy with a) applications running on devices associated to the home environments b) applications running in data centers or c) applications running on another proxies. Common facilities will ease access from applications to storage resources located at cloud proxies (and maybe shared across several cloud proxy devices) or to some cloud proxy device capabilities (camera, sensors, etc if any). These middleware and common facility technologies will be defined and developed as part of the I2ND (Interface to Network and Devices) chapter.

3.2.5.3 *Critical product attributes*

- Cloud proxy as evolution of the home hub, able to federate the connected devices and expose functionalities to support a large variety of service bundles
- Capability to host applications and govern allocation of computational resources beyond centralized data centers.

3.2.6 Resource Monitoring

3.2.6.1 *Target usage*

It has been said that "If you can't measure it you can't manage it", and this maxim is as true in computer science as in industrial management. Ability to monitor the different GEs (via exposed metrics) and their interaction is an absolute requirement in order to support SLOs and SLAs, as well as investigate causes and pin responsibility for SLO infringements. The same goes for the infrastructure of the cloud, including hardware, OS and driver setup, host scheduling and resource sharing decisions, etc.

Monitoring data is also indispensable in order to provide dynamic scaling of resource deployment by supervisory processes. Metrics provided by a monitoring system will enable a provider to understand usage of the system and know when to scale a sub-system. Metrics should be offered to users of cloud hosting so that they may build their own self-management systems or as an offer to enable further trust and visibility into their provisioned services/resources.

3.2.6.2 *GE description*

Although monitoring is a crosscutting concern for the GEs, the specific metrics are GE-specific. Each GE would define a set of metrics to be provided by it, some of them raw metrics and some computed from combinations of other data. Whether a metric is raw or computed is an implementation matter for a GE, the user need not know which is which. The figure below outlines the possible implementation of monitoring by a GE.

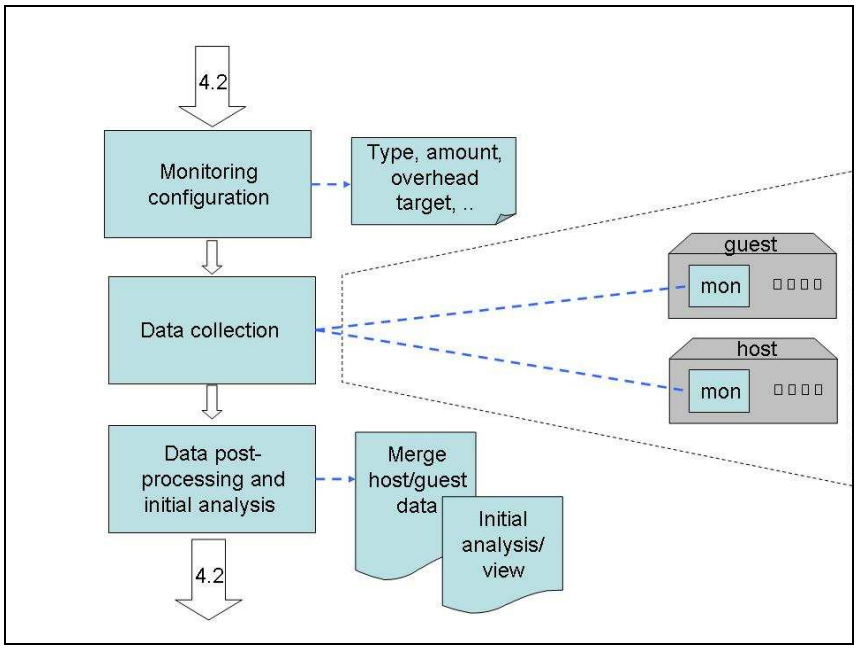


Figure 10 Implementation of monitoring in a GE

Monitoring configurations should include the following dimensions:

- Metrics to monitor
- Metric configuration
- Granularity of monitoring
- Overhead targets
- Distribution mode (push/pull) for the collected data

Data collection is very much dependent on the specific GE and metric.

The work of monitoring is not finished once the data is collected. The data actually collected may need to be post-processed before being provided to its users. User-defined metrics should be offered. This calls for supporting services (e.g., CEP). A storage system is needed to store the metrics. Additionally, data retention rules may apply to some of the data, for business or legal purposes in order to be available for audits or failure analysis.

The collected data needs to be distributed to interested parties using a publish/subscribe messaging system supporting both push and pull protocols. Metrics should only be distributed in a push fashion when a user or system set threshold is reached. These thresholds should be simple and let further processing of the metric data done by subscribers.

Finally, since the GE providers are the best in interpreting the GE's monitoring data, a Monitoring and Metering GE may go one step further and provide additional analysis of the collected data, highlighting performance anti-patterns in the data or identifying high-level effects that may follow from the observed phenomena.

3.3 Question Marks

We list hereafter a number of questions that remain open. Further discussion on these questions will take place in the coming months.

3.3.1 Security aspects

Very much like monitoring, security is a cross cutting concern of all generic enablers and components within the FI-WARE Cloud Hosting chapter. Whereas monitoring mainly deals with aspects "of the moment", security has additional dimensions to consider namely pre- and post- execution of any action initiated by actors of the system. The common areas to consider aspects of security are:

- Authentication (AuthN): This is one of the pre-execution dimensions. Users must first be authenticated to carry out various operations upon the resources and services that the cloud hosting chapter will offer. For efficient operations in a provider who may have multiple but related services, this authorisation should be of the single-sign-on type.
- Access/Authorization (AuthZ): Again this is another pre-execution dimension. Although a user may be authenticated, they may not have the credential to carry out certain actions.
- Audit & Accounting: This dimension is most closely related to monitoring. Audit and accounting operations are taking when decisions and actions are made throughout the lifecycle of a service and form a historical record of all operations taken upon services. By exposing as much auditing and monitoring information to the client, that client will be imbued with greater trust in the service provider.

Related to post-execution dimensions these could be seen more related to the decommissioning policies that a provider enforces. An example of such a decommissioning policy might be all previous machine images are securely wiped or another may be that all audit and accounting logs must be kept for a specific period of time after decommissioning. As a result it could be required that policy GEs be required to enable such policy enforcement.

Other prescient questions that need resolution are:

- What Security GEs are suitable for integration within the Cloud Hosting chapter?
- How should this integration occur? What APIs and transports are offered?
- Are/will those GE's interchangable with systems that may be specific and linked to the Cloud Hosting chapter?

3.3.2 Other topics still under discussion

Following is a list of some questions still to be address. They are listed here so that the reader finds out that they haven’t been ignored.

- What value-add services beyond basic PaaS (and Object Storage) capability do we plan to provide? E.g., DB, messaging/queuing, etc.
- What level of integration do we envision between the "Cloud Edge" GE and other GEs?
- What is the exact division of labor and interfaces between the Cloud Edge Resource Management GE in this chapter, and the Cloud Edge GE in the I2ND chapter?
- What capabilities of I2ND GE, particularly NetIC or S3C Ges, will be leveraged by our service management layer?
- What monitoring capabilities do we need to provide, end-to-end?
- Should we prioritise horizontal scaling (the “cloud” way) over vertical scaling?

- What is the exact scope of what we are going to provide in the area of Metering, Accounting and Billing, and what capabilities will we (or the Cloud Hosting users) be able to leverage from the Business Framework provided by the Apps/Services Ecosystem and Delivery chapter?
- There could be the need (e.g. from monitoring) for a complex event processing system. Can we use the CPE GE from another the Data/Context Management chapter (presumably yes)?
- It seems that there will be the necessity for messaging –based communication. Is a single technology going to be used all across FI-WARE?

3.4 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)

- **Infrastructure as a Service** (**IaaS**) -- a model of delivering general-purpose **virtual machines** (**VMs**) and associated resources (CPU, memory, disk space, network connectivity) on-demand, typically via a self-service interface and following a pay-per-use pricing model. The virtual machines can be directly accessed and used by the IaaS consumer (e.g., an application developer, an IT provider or a service provider), to easily deploy and manage arbitrary software stacks.
- **Platform as a Service** (**PaaS**) -- an application delivery model in which the clients, typically application developers, follow a specific programming model to develop their applications and or application components and then deploy them in hosted runtime environments. This model enables fast development and deployment of new applications and components.
- **Virtual Appliances** (**vApp**, also referred to as "**service**") -- pre-built software solutions, comprised of one or more Virtual Machines that are packaged, updated, maintained and managed as a unit. Virtual appliances are typically packaged in an Open Virtualization Format (**OVF**), developed by Distributed Management Task Force (DMTF) standardization body.
- **Key Performance Indicators** (**KPIs**) -- quantifiable metrics reflecting the level of offered service with respect to specific non-functional requirements such as performance, availability, resiliency, etc. KPI is usually computed as a function of one or more low level **metrics** – analytical, quantative measurements intended to quantify the state of a process, service or system. KPIs may relate either to the long term measures of the service level where raw metrics are averaged and summarized over a long time scale to guide strategic decisions about the service provisioning, or short term measures of service level, triggering proactive optimization.
- **Service Elasticity** is the capability of the hosting infrastructure to scale a service up and down on demand. There are two types of elasticity -- **vertical** (typically of a single VM), implying the ability to add or remove resources to a running VM instance, and **horizontal** (typically of a clustered multi-VM service), implying the ability to add or remove instances to/from an application cluster, on-demand. Elasticity can be triggered manually by the user, or via an **Auto-Scaling** framework, providing the capability to define and enforce automated elasticity policies based on application-specific KPIs.
- **Service Level Agreement** (**SLA**) is legally binding contract between a service provider and a service consumer specifying terms and conditions of service provisioning and consumption. Specific SLA clauses, called **Service Level Objectives** (**SLOs**), define non-functional aspects of service provisioning such as performance, resiliency, high availability, security, maintenance, etc. SLA also specifies the agreed upon means for verifying SLA compliance, customer compensation plan that should be put in effect in case of SLA incompliance, and temporal framework that defines validity of the contract.
- **Cloud Proxy** devices are located outside the Cloud. May correspond to end-user devices (any device the user may use to interact with cloud applications like a PC or a tablet, but also sensors, displays…)

or to a more complex structure like a home gateway, i.e., a special device located in the home network. Cloud proxies provide the ability to host applications or use storage resources located closer to the end user, intended to provide an improved user experience.

- **CDMI**. The Cloud Data Management Interface (CDMI) defines the functional interface that applications will use to create, retrieve, update and delete (CRUD) data elements from the Cloud defined by the Storage Networking Industry Association (SNIA) group
- **Cloud Service Management API** -- a RESTfull, resource-oriented API accessed via HTTP which uses XML-based representations for information interchange and allows deployment of OVF-based service manifests or manifest fragments (thus enabling incremental deployment). It supports extensions to the Open Virtualization Format (OVF) in order to support advanced Cloud capabilities and is based on the vCloud specification (published by VMware) and submitted to the DMTF for consideration.
- **OCCI** is a protocol and API for the management of cloud service resources. It comprises a set of open community-lead specifications delivered through the Open Grid Forum. OCCI was originally initiated to create a remote management API for IaaS model based Services. It has since evolved into a flexible API with a strong focus on integration, portability, interoperability and innovation while still offering a high degree of extensibility.

3.5 References

[VMWare 09]	VMWare. vCloud API Programming Guide, Version 0.8.0. Online resource., 2009. http://communities.vmware.com/static/vcloudapi/vCloud
[TCloud 10]	API Programming Guide v0.8.pdf. Telefónica. TCloud API Specification, Version 0.9.0. Online resource., 2010. http://www.tid.es/files/doc/apis/TCloud API Spec v0.9.pdf.
[OVF 08]	DMTF. Open virtualization format specification. Specification DSP0243 v1.0.0d. Technical report, Distributed Management Task Force, Sep 2008. https://www.coin-or.org/OS/publications/optimizationServicesFramework2008.pdf
[TCloudServer]	Tcloud-server implementation, http://claudia.morfeo-project.org/wiki/index.php/TCloud_Server
[ServiceManifest 10]	DMTF. the distributed management task force webpage. Online resource.,2010. http://www.dmtf.org. Service manifest definition DMTF's OVF
[Cáceres et al. 10]	J. Cáceres, L. M. Vaquero, L. Rodero-Merino, A. Polo, and J. J. Hierro. Service Scalability over the Cloud. In B. Furht and A. Escalante, editors, Handbook of Cloud Computing, pages 357–377. Springer US, 2010. 10.1007/978-1-4419-6524-0 15.
[EC2SLA]	Amazon EC2 SLA http://aws.amazon.com/ec2-sla/
[S3SLA]	Amazon S3 SLA http://aws.amazon.com/ec2-sla/

[GoGridSLA]	GoGrid SLA http://www.gogrid.com/legal/sla.php
[RackspaceSLA]	Rackspace SLA http://www.rackspace.com/cloud/legal/sla/
[GoogleAppsSLA]	Google Apps SLA http://www.google.com/apps/intl/en/terms/sla.html
[AzureSLA]	Microsoft Azure SLA http://www.microsoft.com/windowsazure/sla/
[Reservoir-Computer2011]	Benny Rochwerger, David Breitgand, A. Epstein, D. Hadas, I. Loy, Kenneth Nagin, J. Tordsson, C. Ragusa, M. Villari, Stuart Clayman, Eliezer Levy, A. Maraschini, Philippe Massonet, H. Muñoz, G. Tofetti: Reservoir - When One Cloud Is Not Enough. IEEE Computer 44(3): 44-51 (2011)
[Reservoir-Architecture2009]	Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén S. Montero, Yaron Wolfsthal, Erik Elmroth, Juan A. Cáceres, Muli Ben-Yehuda, Wolfgang Emmerich, Fermín Galán: The Reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development 53(4): 4 (2009)
SchadDJQ-VLDB10	Jörg Schad, Jens Dittrich, Jorge-Arnulfo, Quiané-Ruiz, Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance, In Proceedings of The Vldb Endowment, Vol 3, pages 460—471, 2010}
DejunPC-2011	Jiang Dejun, Guillaume Pierre, Chi-Hung Chi, Resource Provisioning of Web Applications in Heterogeneous Clouds, in Proceedings of the 2nd USENIX Conference on Web Application Development, 2011
VMware-CloudArch1.0	Vmware, Architecting a vCloud, Technical Paper, version 1.0
AppSpeed	VMware vCenter AppSpeed User's Guide AppSpeed Server 1.5
CloudFormation	Amazon EC2 Cloud Formation http://aws.amazon.com/cloudformation/
BenYehuda-ICAC2009	Muli Ben-Yehuda, David Breitgand, Michael Factor, Hillel Kolodner, Valentin Kravtsov, Dan Pelleg: NAP: a building block for remediating performance bottlenecks via black box network analysis. ICAC 2009: 179-188
[Chapman 10]	C. Chapman, W. Emmerich, F. G. Márquez, S. Clayman, and A. Galis. Software architecture definition for on-demand cloud provisioning. In HPDC '10: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pages 61–72, New York, NY, USA, 2010. ACM.

[CDMI 11]	Cloud Data Management Interface Version, Working Draft, version 1.0.1h, March 30, 2011. http://cdmi.sniacloud.com/
[REST]	"Representational State Transfer" - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[RESTful Web 07]	Richardson, Leonard and Sam Ruby, RESTful Web Services, O'Reilly, 2007.
[OCCI_GCDM]	A. Edmonds, T. Metsch, and A. Papaspyrou, "Open Cloud Computing Interface in Data Management-related Setups," Springer Grid and Cloud Database Management, pp. 1–27, Jul. 2011.
[Vaquero et al. 11]	L. M. Vaquero, J. Caceres and D. Morán, The Challenge of Service Level Scalability for the Cloud, International Journal of Cloud Applications and Computing, Volume 1, Number 1, 2011, pp 34-44

4 Data/Context Management

4.1 Overview

The availablity of advanced platform functionalities dealing with gathering, processing, interchange and exploitation of data at large scale is going to be cornerstone in the development of intelligent, customized, personalized, context-aware and enriched application and services beyond those available on the current Internet. These functionalities will foster the creation of new business models and opportunities which FI-WARE should be able to capture.

Data in FI-WARE refers to information that is produced, generated, collected or observed that may be relevant for processing, carrying out further analysis and knowledge extraction. It has associated a **data type** and a **value**. FI-WARE will support a set of built-in **basic data types** similar to those existing in most programming languages. Values linked to basic data types supported in FI-WARE are referred as **basic data values**. As an example, basic data values like '2', '7' or '365' belong to the integer basic data type.

A **data element** refers to data whose value is defined as consisting of a sequence of one or more <name, type, value> triplets referred as **data element attributes**, where the type and value of each attribute is either mapped to a basic data type and a basic data value or mapped to the data type and value of another data element. Note that each data element has an associated data type in this formalism. This data type determines what concrete sequence of attributes characterizes the data element.

There may be **meta-data** (also referred as semantic data) linked to attributes in a data element. However, existence of meta-data linked to a data element attribute is optional.

Applications may assign an identifier to data elements in order to store them in a given **Data Storage**, e.g. a Data Base. Such identifier will not be considered part of the structure of the data element and the way it can be generated is out of the scope of this specification. Note that a given application may decide to use the value of some attribute linked to a data element as its identifier in a given Data Storage but, again, there is no identifier associated to the representation of a data element.

The structure associated to a **data element** is represented in Figure 11.

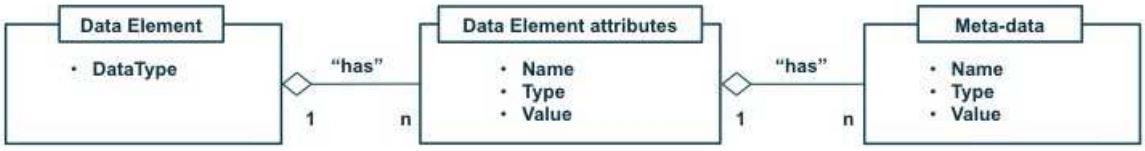


Figure 11 Data Element Structure

A cornerstone concept in FI-WARE is that data elements are not bound to a specific format representation. They can be represented as an XML document at some point and then translated into another XML document representation later on, or marshaled as part of a binary message being transferred. Data elements can be stored e.g. in a Relational Database, in an RDF Repository or as entries in a noSQL data base like MongoDB, adopting a particular storage format that may be the same or different respectively to the format used for their transfer. It should be possible to infer the data type of a given data element based on the XML document or on the transferred message format (e.g., by a specific element of the XML document if the same XML style and encoding is used to represent data elements of different types or by a specific used XML style) or based on the specific storage structure format used to store it (e.g., may be inferred from the name of the table in which the data element is stored).

The way data elements are represented in memory by FI-WARE Data/Context Generic Enablers is not specified. Therefore, the implementer of a FI-WARE Datat/Context Generic Enabler may decide the way data elements are represented in memory.

Context in FI-WARE is represented through **context elements**. A context element extends the concept of **data element** by associating an EntityId and EntityType to it, uniquely identifying the entity (which in turn may map to a group of entities) in the FI-WARE system to which the context element information refers. In addition, there may be some attributes as well as meta-data associated to attributes that we may define as mandatory for context elements as compared to data elements.

Context elements are typically created containing the value of attributes characterizing a given entity at a given moment. As an example, a context element may contain values of the "last measured temperature", "square meters" and "wall color" attributes associated to a room in a building.

Note that there might be many different context elements referring to the same entity in a system, each containing the values of a different set of attributes. This allows that different applications handle different context elements for the same entity, each containing only those attributes of that entity relevant to the corresponding application. It will also allow representing updates on the set of attributes linked to a given entity: each of these updates can actually take the form of a context element and contain only the value of those attributes that have changed.

The structure of a context element is represented in Figure 12.

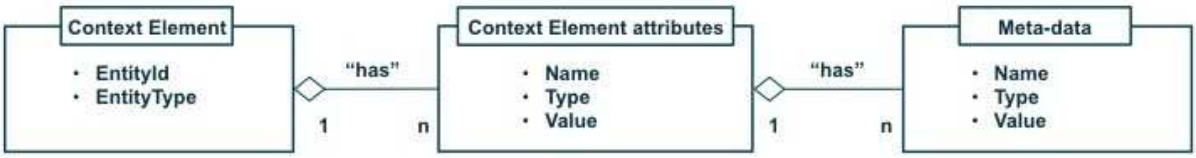


Figure 12 Context Element Structure Model

Note that all the statements made with respect to data elements in the previous section would also apply to context elements.

An **event** is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. Events typically lead to creation of some data or context element, thus enabling that information describing or related to events be handled by applications or event-aware FI-WARE GEs (e.g., the Publish/Subscribe Broker GE, when handling update/notifications, or the CEP GE). As an example, a sensor device may be measuring the temperature and pressure of a given boiler, sending a context element every five minutes associated to that entity (the boiler) that includes the value of these to attributes (temperature and pressure) or just the one that has changed. The creation and sending of the context element is an event, i.e., something that has occurred (the sensor device has sent new measures). As another example, a mobile handset may export attributes like "Operating System" or "Screen size". A given application may query for the value of these two attributes in order to adapt the content to be delivered to the device. As a result, the mobile handset creates and replies a context element back to the application. This response may be considered as well an event, i.e., something that has occurred (the mobile handset has replied to a request issued by an application).

Since the data/context elements that are generated linked to an event are the way events get visible in a computing system, it is common to refer to data/context elements related to events simply as "events" while describing the features of, or the interaction with, event-aware FI-WARE GEs. For convenience, we also may use the terms "data event" and "context event". A "data event" refers to an event leading to creation of a data element, while a "context event" refers to an event leading to creation of a context element.

The word **event object** is used to mean a programming entity that represents such an occurrence (event) in a computing system [EPIA]. Events are represented as event objects within computing systems to

distinguish them from other types of objects and to perform operations on them, also known as **event processing**.

In FI-WARE, event objects are created internally to some GEs like the Complex Event Processing GE or the Publish/Subscribe Broker GE. These event objects are defined as a data element (or a context element) representing an event to which a number of standard event object properties (similar to a header) are associated internally. These standard event object properties support certain event processing functions. The concrete set of standard event object properties in FI-WARE is still to be defined but we may anticipate that one of these properties would be the time at which the event object is detected by the GE (arrives to the GE). This will, for example, allow to support functions that can operate on events that exceed certain age in the system. Tools will be provided enabling applications or admin users to assign values to those event object properties based on values of data element attributes (e.g., source of the event or actual capture time). An event object may wrap different characteristics of the data element (i.e., DataType) or the context element (i.e., EntityId and EntityType).

Unless otherwise specified, all staments in the rest of the chapter that make use of the term "data (element)" also apply to context (element) as well as to events, related or not to data/context elements.

Figure 13 presents a high-level view of the Reference Architecture of the Data/Context Management chapter in FI-WARE. Defined GEs (marked in light blue in the figure) can be instantiated in a flexible and coherent way, enabling different FI-WARE Instances to pick and configure different GEs according to demands and requirements of applications that will run on top. Following is a brief description of them:

- **Publish/Subscribe Broker GE**, that allows applications to interchange heterogeneous events following a standard publish – subscribe paradigm.
- **Complex Event Processing GE**, which has to do with the processing of event streams in real-time that will generate immediate insight, enabling applications to instantly response to changing conditions on certain customers or objects (entities such as devices, applications, systems, etc.).
- **BigData Analysis GE**: which enables to perform a map-reduce analysis of large amount of data both on the go or previously stored.
- **Multimedia Analysis Generation GE**, which performs the automatic or semiautomatic extraction of meta-information (knowledge) based on the analysis of multimedia content.
- **Unstructured data analysis GE**, which enables the extraction of meta-data based on the analysis of unnstructured information obtained from web resources.
- **Meta-data pre-processing GE**, which ease the generation of programming objects from several metadata-formats.
- **Localization GE**, which provides geo-location information as a context information obtained from devices.
- **Query Broker GE**, which deals with the problem of providing a uniform query access mechanism for retrieval of data stored in heterogeneous formats.
- **Semantic Annotation GE**, which allows to enrich multimedia information or other data with semantic meta-data tags to be exploited by semantic web applications.
- **Semantic Application Support GE**, which provides support for the core set of semantic web functionalities that ease programming of semantic web applications.

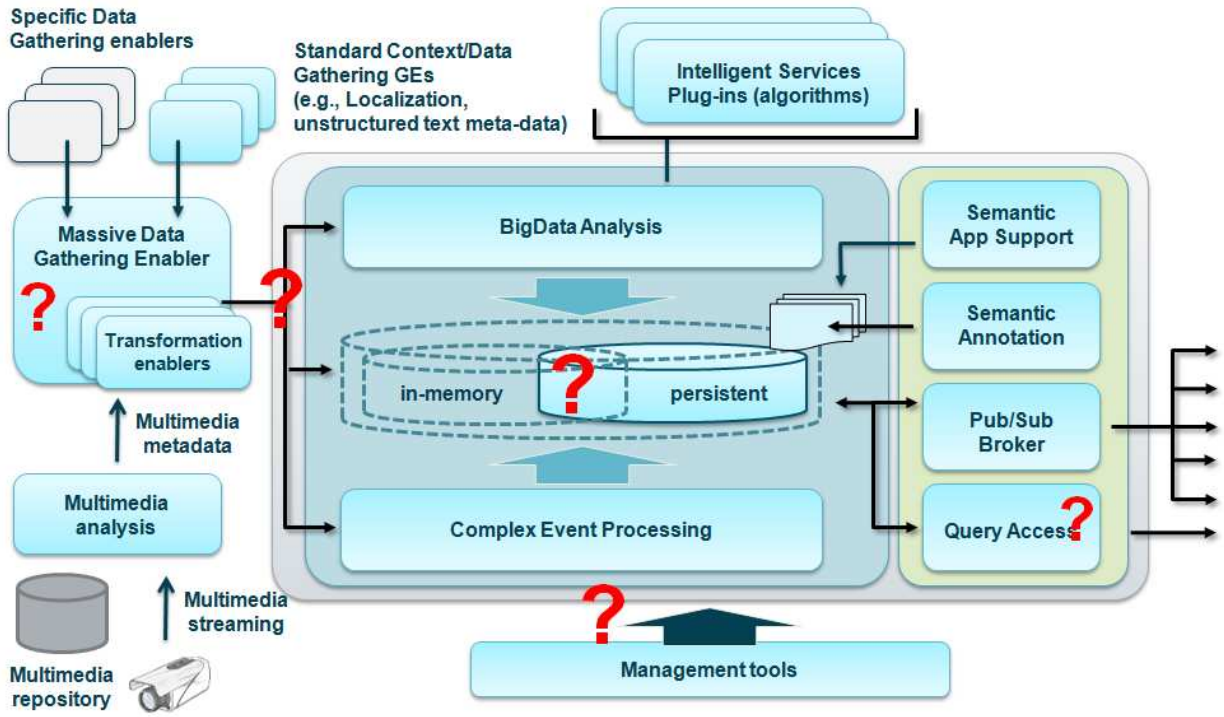


Figure 13 High-Level view of GEs in the Data/Context Management chapter

In addition to these GEs, specific enablers implementing intelligent services have been defined. These enablers interact with off-line and real time processing enablers, as well as with data both in memory or persistent, to provide analytical and algorithmic capabilities in the following areas:

- Social Network Analysis
- Mobility and Behaviour Analysis
- Real-time recommendations
- Behavioural and Web profiling
- Opinion mining

The following sections describe these Generic Enablers (GE) in more detail. Figure 13 displays some question marks that are reviewed at the end of this chapter.

4.2 Generic Enablers

4.2.1 Publish/Subscribe Broker

4.2.1.1 *Target usage*

The Publish/Subscribe Broker is a GE of the FI-WARE platform that enables publication of events by entities, referred as Event Producers, so that published events becomes available to other entities, referred as Event Consumers, which are interested in processing the published events[10]. Applications or even other GEs in the FI-WARE platform may play the role of Event Producers, Event Consumers or both.

A fundamental principle supported by this GE is that of achieving a total decoupling between Event Producers and Event Consumers. On one hand, this means that Event Producers publish data without knowing which Event Consumers will consume published data; therefore they don't need to be connected to them. On the other hand, Event Consumers consume data of their interest, without this meaning they know which Event Producer has published a particular event: they are just interested in the event itself but not in who generated it.

4.2.1.2 *GE description*

The conceptual model and set of interfaces defined for the Publish/Subscribe Broker GE are aligned with the technical specifications of the NGSI-10 interface, which is one of the interfaces associated to Context Management Functions in the Next Generation Service Interfaces (NGSI) defined by the Open Mobile Alliance [OMA-TS-NGSI-Context]. Specifications of the Publish/Subscribe Broker GE in FI-WARE will not support all the interfaces and operations defined for Context Management Functions in NGSI: it will just focus on the NGSI-10 interface and even only those parts of the specifications associated to this interface that are considered most useful to support development of applications in the Future Internet. On the other hand, it will extend the scope of NGSI-10 specifications as to be able to deal with data elements, not just context elements.

Figure 14 illustrates the basic operations that Event Producers and Consumers can invoke on Publish/Subscribe Broker GEs in order to interact with them. Event Producers publish events by invoking the update operation on a Publish/Subscribe Broker GE. Note that when used to publish context elements representing updates on the values of attributes linked to existing entities, only the value of attributes that have changed may be passed. Despite not currently included in the NGSI-10 specifications, Event Producers in FI-WARE also export the query operation, enabling a Publish/Subscriber Broker GE to pull for events, as illustrated in Figure 14.

[10] As explained earlier by "event" we are here indeed referring to data or context elements describing or related to an event.

The Publish/Subscribe Broker GE exports interfaces enabling Event Consumers to consume events in two basic modes:

- Request/response mode, enabling retrieval of events as response to query requests on the Publish/Subscribe Broker GE issued by Event Consumers;
- Subscription mode, enabling to setup the conditions under which events will be pushed to a given Event Consumers. The Publish/Subscribe Broker GE will invoke the notify operation exported by the Event Consumer every time an event fulfilling the condition established in its subscription arrives. Subscriptions may be setup by third applications and not necessarily the Event Consumer itself, as illustrated in the figure. An expiration time can be defined for each subscription by means of invoking operations exported by the Publish/Subscribe Broker GE. If the expiration time is missing, then default values may be used.

When an Event Consumer or a Publish/Subscribe Broker formulates a query, it may formulate it relative to attributes of a given entity (or group of entities). The context elements being sent as response are considered as events.

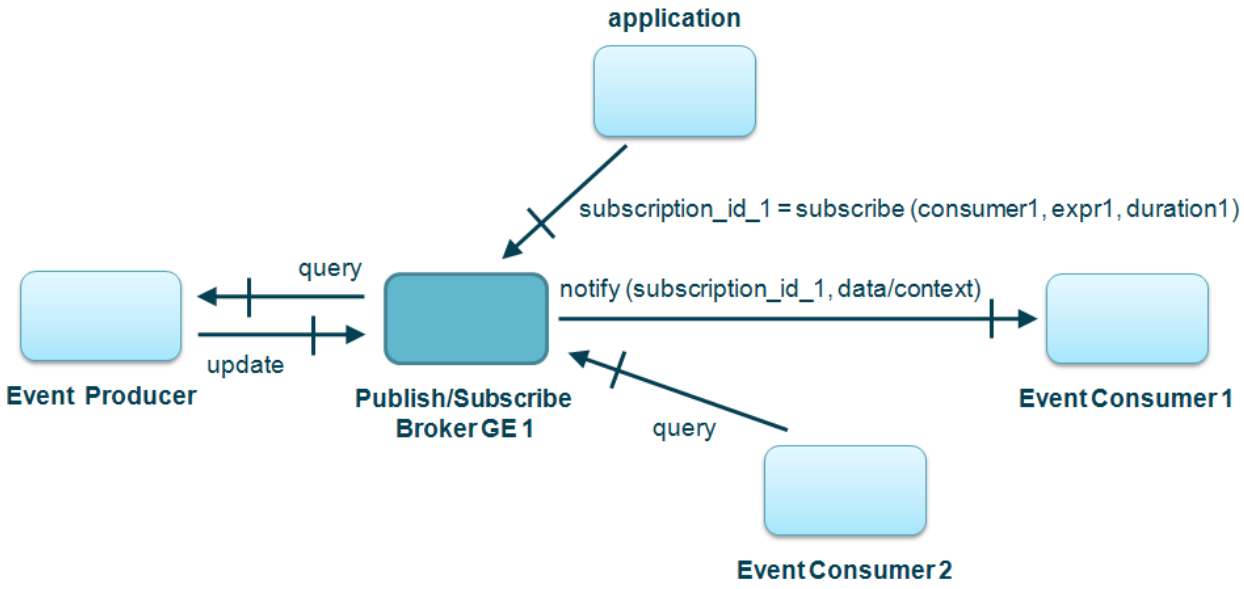


Figure 14 Basic interaction with the Publish/Subscribe Broker GE

Note that events are kept in memory by the Publish/Subscribe Broker GE while events do not exceed a given expiration time. Therefore, they are not dropped from the Publish/Subscribe Broker GE persistent memory/storage just because the result of a query (request/response mode) returns a copy of them. That is, the same query on a Publish/Subscribe Broker GE will return the same events (provided they haven't expired). However, events are notified just once to subscribed Event Consumers in the subscription mode.

Data/context elements associated to events could be of any type. Besides, Event Producers could be either components integrated and embedded into a FI-WARE GE or being part of an application as far as they respect the described interfaces. Therefore, data/context elements in events could be of any provenience and of any granularity such as both raw data (not processed yet through any FI-WARE GE) or a higher level of data abstraction (e.g., insights extracted from the raw data by means of some of the GEs described within this document and shown in Figure 13).

An entity playing the role of Event Producer is an entity that publishes events in this model but could also play the role of an Event Consumer, e.g. may consume events and publish other events based on the consumed ones. Besides, the Publish/Subscribe Broker GE may be connected to many Event Producers so that, the number of Event Producers as well as their availability shall be hidden to the Event Consumer. The Publish/Subscribe Broker GE acts as a intermediate single access point to multiple Event Consumers

for accessing events published by multiple Event Producers. An example of an architecture involving multiple Event Producers and Consumers connected to a Publish/Subscribe Broker GE is shown in Figure 15 .

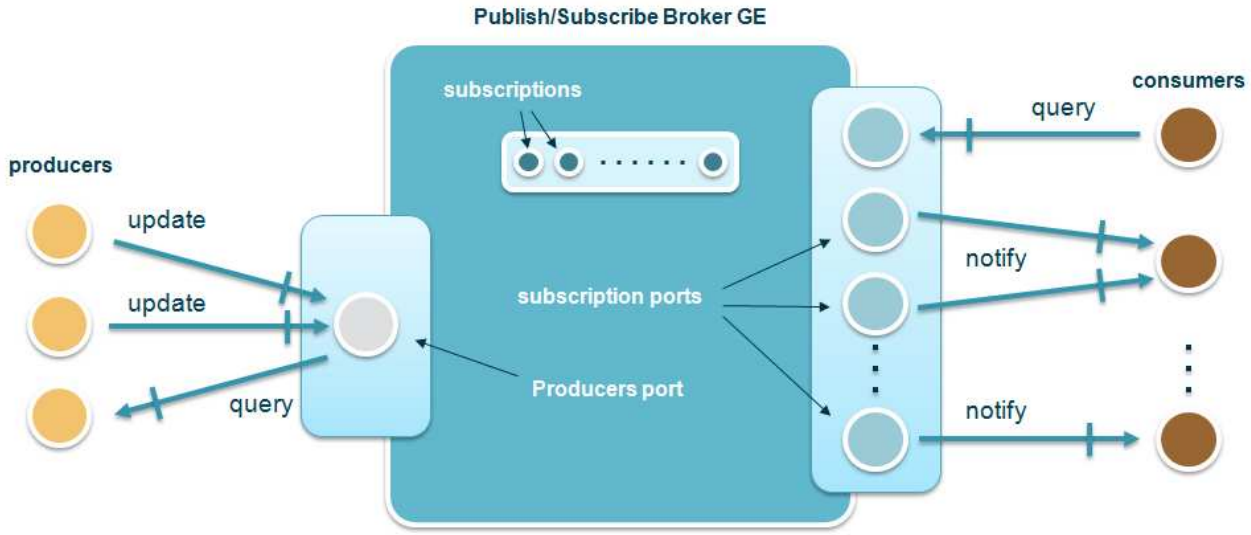


Figure 15 Multiple Event Producers and Consumers connected to a Publish/Subscribe Broker GE

A Publish/Subscribe Broker GE may play the role of Event Consumers, therefore being able to consume events published by another Publish/Subscribe Broker GE (in either of the two existing modes). This will support the ability to federate Publish/Subscribe Broker GEs, even in the case that their implementation is different. This will be useful in scenarios like the management of the Internet of Things, where there might be Publish/Subscribe Broker GEs running in devices, IoT gateways or centralized. A service for discovering Publish/Subscribe Broker GEs based on criteria will be defined in FI-WARE, making it easier to support these highly distributed scenarios involving multiple Publish/Subscribe Broker GEs.

4.2.1.3 *Critical product attributes*

- All data types (generic data or context elements) are available for consumption through the same Publish/Subscribe GE interface
- Comprehensive publish/subscribe interfaces enabling to formulate powerful subscription conditions
- Simple publish/subscribe interface enabling easy and fast integration with consumer applications following a pull or push style of communication
- OMA-based standard interfaces, easing the interworking with many devices while still being useful for event publication/subscription at backend systems
- Ability to develop light and efficient implementations (capable to address real- or near real-time delivery of events and to run on small devices)
- Extendible (ability to extend interfaces or add interfaces in order to introduce new or domain-specific features)
- Scalable (enabled through Publish/Subscribe Broker GE federation)
- Reserved facility to be auto-cleaning and self-controlled (to purge unused or forgotten subscriptions)

4.2.2 Complex Event Processing

4.2.2.1 *Target usage*

Complex Event Processing (CEP) is the analysis of event data in real-time to generate immediate insight and enable instant response to changing conditions. Some functional requirements this technology addresses include event-based routing, observation, monitoring and event correlation. The technology and implementations of CEP provide means to expressively and flexibly define and maintain the event processing logic of the application, and in runtime it is designed to meet all the functional and non-functional requirements without taking a toll on the application performance, removing one issue from the application developer's and system managers concerns.

For the primary user of the real-time processing generic enabler, namely the consumer of the information generated, the Complex Event Processing GE (CEP GE) addresses the user's concerns of receiving the relevant events at the relevant time with the relevant data in a consumable format. Relevant - meaning of relevance to the consumer\subscriber to react or make use of the event appropriately. Figure 16 depicts this role through a pseudo API *derivedEvent(type,payload)* by which, at the very least, an event object is received with the name of the event, derived out of the processing of other events, and its payload.

The designer of the event processing logic is responsible for creating event specifications and definitions (including where to receive them) from the data gathered by the Massive Data Gathering Generic Enabler. The designer should also be able to discover and understand existing event definitions. Therefore FI-WARE, in providing an implementation of a Real-time CEP GE, will also provide the tools for the designer. In addition, APIs will be provided to allow generation of event definitions and instructions for operations on these events programmatically, such as by an application or by other tools for other programming models that require Complex Event Processing such as the orchestration of several applications into a composed application using some event processing. In Figure 16 these roles are described as Designer and Programs making use of the pseudo API *deploy definitions/instructions*.

Finally, the CEP GE addresses the needs of an event system manager and operator, could be either real people or management components, by allowing for configurations (such as security adjustments), exposing processing performance, handling problems, and monitoring the system's health, represented in Figure 16 as Management role making use of the pseudo API *configuration/tuning/monitoring*.

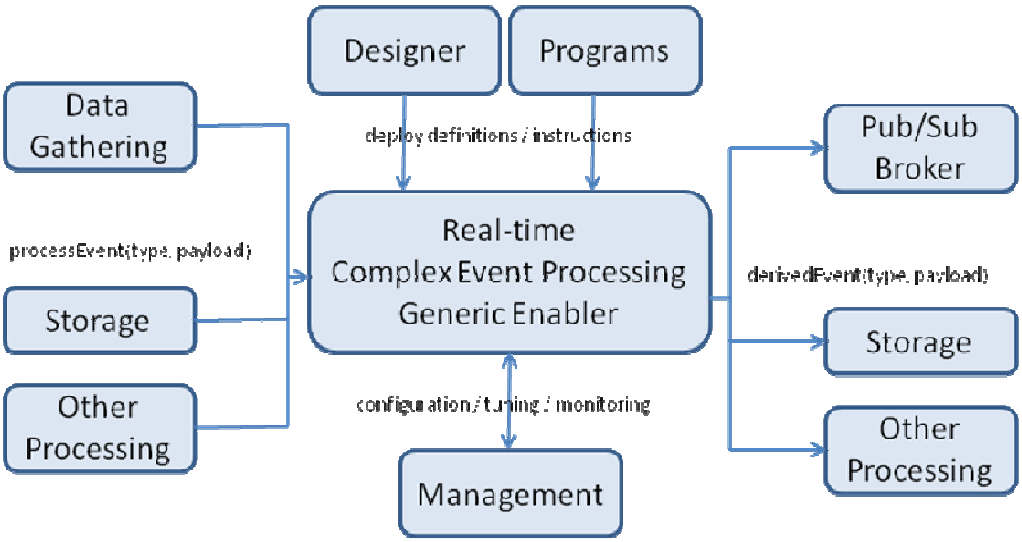


Figure 16 Interactions with and APIs of the Real-time CEP Generic Enabler

4.2.2.2 *GE description*

The Complex Event Processing Generic Enabler (CEP GE) provides:

- tools to define event processing applications on data interpreted as events, either manually or programmatically
- execution of the event processing application on events as they occur and generation of derived events accordingly
- management of the runtime

The functions supported by such a GE aligns with the functional architecture view produced by the Reference Architecture Work Group of the Event Processing Technical Society (EPTS) [EPTS][EPTS-RA 10], depicted in Figure 17 and further elaborated bellow.

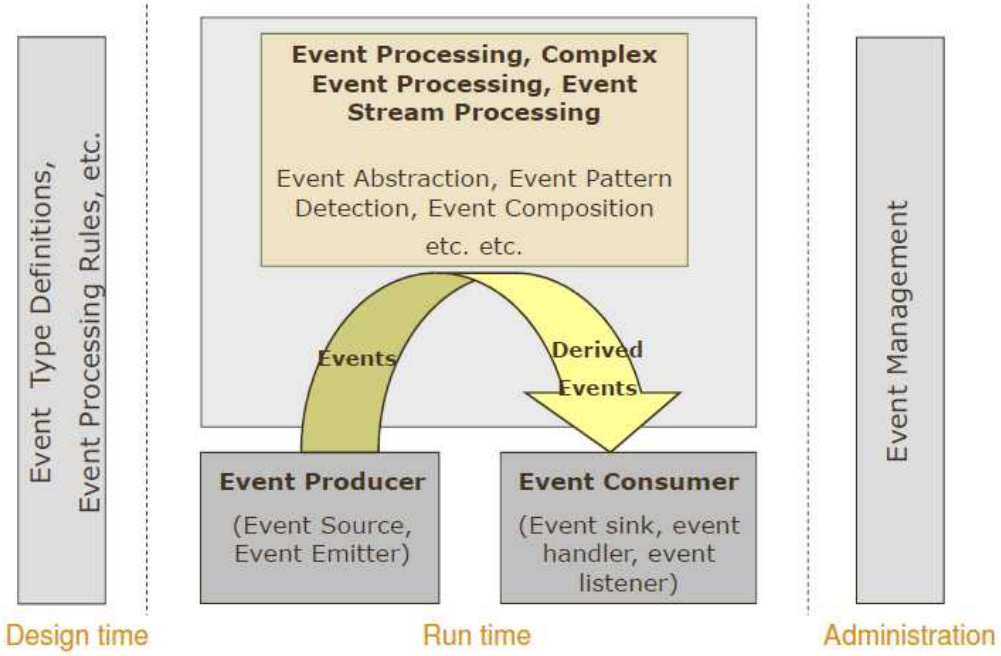


Figure 17 Functional View of Event Processing

Entities connected to the CEP GE (application entities or some other GEs like the Publish/Subscribe Broker GE) can play two different roles: the role of **Event Producer** or the role of **Event Consumers**. Note that nothing precludes that a given entity plays both roles.

Event Producers are the source of events for event processing. They can provide events in two modes:

- "Push" mode: The Event Producers push events into the Complex Event Processing GE by means of invoking a standard operation the GE exports.
- ”Pull” mode: The Event Producer exports a standard operation that the Complex Event Processing GE can invoke to retrieve events.

Event Consumers are the sink point of events. Following are some examples of event consumers:

- Dashboard: a type of event consumer that displays alarms defined when certain conditions hold on events related to some user community or produced by a number of devices..
- Handling process: a type of event consumer that consumes processed events and performs a concrete action.
- The Publish/Subscribe Broker GE (see section 4.2.1): a type of event consumer that forwards the events it consumes to all interested applications based on a subscription model.

Despite not being decided yet, it is most likely the case that Event Producers and Event Consumers that can be connected to the CEP GE in FI-WARE export the interfaces of Event Producers and Consumers as specified in section 4.2.1 (description of the Publish/Subscribe Broker GE). This will allow that a Publish/Subscribe Broker GE is connected to forward events to a CEP GE or, viceversa, a CEP GE forward events result of processing to a Publish/Subscribe Broker GE.

The CEP GE in FI-WARE implements event processing functions based on the design and execution of Event Processing Networks (EPN). Processing nodes that make up this network are called Event Processing Agents (EPAs) as described in the book "Event Processing in Action" [EPIA]. The network describes the flow of events originating at event producers and flowing through various event processing agents to eventually reach event consumers, see Figure 18 for an illustration. Here we see that events from Producer 1 are processed by Agent 1. Events derived by Agent 1 are of interest to Consumer 1 but are also processed by Agent 3 together with events derived by Agent 2. Note that the intermediary processing between producers and consumers in every installation is made up of several functions and often the same function is applied to different events for different purposes at different stages of the processing. The EPN approach allows to deal with this in an efficient manner, because a given agent may receive events from different sources. At runtime, this approach also allows for a flexible allocation of agents in physical computing nodes as the entire event processing application can be executed as a single runtime artifact, such as Agent 1 and Agent 2 in Node 1 in Figure 18, or as multiple runtime artifacts according to the individual agents that make up the network, such as Agent 1 and Agent 3 running within different nodes. Thus scale, performance and optimization requirements may be addressed by design. The reasons for running pieces of the network in different nodes or environments vary, for example:

- Distributing the processing power
- Distributing for geographical reasons – process as close to the source as possible for lower networking
- Optimized and specialized processors that deal with specific event processing logic

Another benefit in representing event processing applications as networks is that entire networks can be nested as agents in other networks allowing for reuse and composition of existing event processing applications.

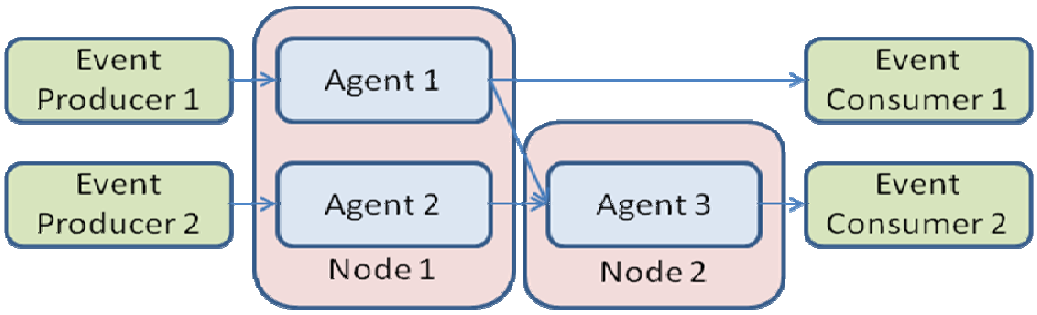


Figure 18 Illustration of an Event Processing Network made of producers, agents and consumers

The event processing agents and their assembly into a network is where most of the functions of this GE are implemented. In FI-WARE, behavior of an event processing agent is specified using a rule-oriented language that is inspired by the ECA (Event-Condition-Action) concept and may better be described as Pattern-Condition-Action. Rules in this language will consist in three parts:

- A **pattern detection** that makes a rule of relevance
- A set of **conditions** (logical tests) formulated on events as well as external data
- A set of **actions** to be carried out when all the established conditions are satisfied

Following is an indication of the capabilities to be support in each part of the rule language. A description of how such rules are assembled with simplifications is described in Event Processing Agent in Detail.

Pattern Detection

In the pattern detection part the user may program patterns over **selected events within an event processing context** (such as a time window or segmentation) and only if the pattern is matched the rule is of relevance and according to conditions, the action part is executed. Examples for such patterns are:

- **Sequence**, meaning events need to occur in a specified order for the pattern to be matched
- **Count**, a number of events need to occur for the pattern to be matched

Event Processing Context [EPIA] is defined as a named specification of conditions that groups event instances so that they can be processed in a related way. It assigns each event instance to one or more context partitions. A context may have one or more context dimensions and can give rise to one or more context partitions. Context dimension tells us whether the context is for a temporal, spatial, state-oriented, or segmentation-oriented context, or whether it is a composite context that is to say one made up of other context specifications. Context partition is a set into which event instances have been classified.

Conditions

The user of the CEP GE may program the following kind of conditions in a given rule:

- **Simple conditions**, which are established as predicates defined over single events of a certain type
- **Complex conditions**, which are established as logical operations on predicates defined over a set of events of a certain type:
  - **All** (conjunction) meaning that all defined predicates must be true
  - **Any** (disjunction) meaning that at least one of the defined predicates must be true
  - **Absence** (negation) meaning that none of the defined predicates can be true

Predicates defined on events can be expressed based on a number of predefined operators applicable over:

- values of event data fields
- values of other properties inherent to an event (e.g., lifetime of the event)
- external functions the GE can invoke and to which event data field values or event property values can be passed

Note that the conditions selected for a given rule establish whether processing of that rule is stateless or stateful. Stateless processing requires that the conditions apply to a single event and are only formulated over properties of the event, without relying on external variables. Stateful processing applies to multiple events or even when a single event is processed but some of the conditions rely on external variables. A processing agent is stateless when processing of all rules governing its behavior are stateless. Otherwise, the processing agent is stateful.

Actions

The user of the GE may program the following kind of actions in a given rule:

- **Transformations**, defined over events satisfying the rule, which may consist in generating a new event whose data is the result of:
  - **Projecting** a subset of the data fields from one or several of the events satisfying the rule
  - **Translating** values of projected data fields into new values as a result of applying some programmed function
  - **Enriching** data of the new event with data not present originally

- **Forwarding actions**, which would consist in forwarding one or several of the events satisfying the rule
- **Invocation of external services** that allow achieving some desired effect in the overall system.

Note that several transformations can be programmed in the same rule. This allows an event or set of events to be split in multiple derived events. In addition, external processes being executed as the result of an action may lead to updates of variables based on which certain functions used in predicates of rule conditions are formulated.

Designing EPNs

At design time the functional aspects of this Generic Enabler include the definition, modelling, improvement, and maintenance of the artefacts used in event processing. This is an integration point with FI-WARE Tools GEs through the management tools. These artefacts are:

- event definitions, includes at the very least a type name and in most cases also definition of the payload
- event processing network assembled by event processing agents

These artefacts can be programmatically developed and deployed on the fly to the execution or can be developed by users with form based tools (eclipse or web based) and manually deployed to the execution – also on the fly.

Event Processing Agent in Detail

To simplify the specification of an event process agent, rather than requiring to model the logic associated to the event processing using a rule-oriented language, a framework based on a number of building blocks is provided to assist in specifying the logic. This allows the user to express his intent without being familiar with the logical and temporal operators in the rule-oriented language and without the need to write logical statements that are long and sometime difficult to understand. This is done by specifying three building blocks that make up and agent as well as input and output terminals that specify the type of events to be considered in a rule and the type of events that may be derived by the rule respectively, as depicted in Figure 19.

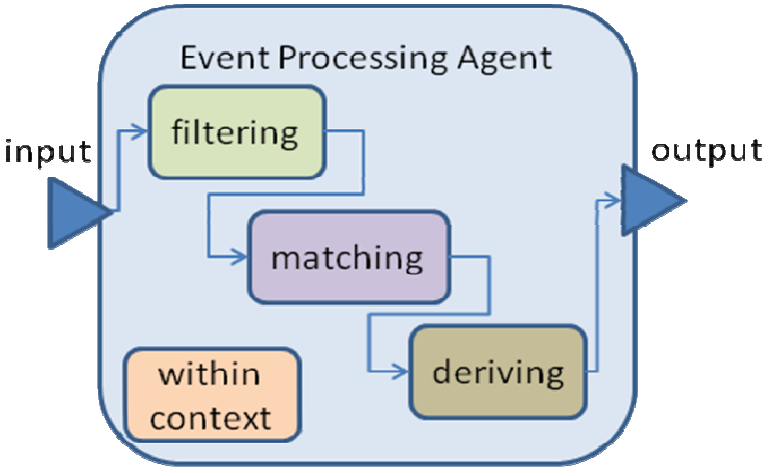


Figure 19 The building block of Event Processing Agent – simpler specification of the logic

Only events of the type that have been specified in the input terminals of the agent will be considered for the rule. Then, the following building blocks are defined:

- The first building block is the filter block, an optional block, where filters can be applied on individual input events whether to consider them or not in further execution of the rule.

- The second building block is the matching block, an optional block, where the events are matched against a specified pattern and where matching sets are created.
- The final building block is the derivation block, the only mandatory block, where matching sets, filtered events or direct input events are used to compose one or more derived events according to conditions and expressions.

The derived events are specified in the output terminals to be selected as inputs by other agents.

This entire rule may be executed in a processing context (temporal, segmentation (group-by), state, space and spatiotemporal) which is also specified for an agent when needed. This is equivalent to the processing context of the pattern part of the rule.

For example, if only the derivation block is specified then the agent provides transformation operations (the type of transformation is dependent on the derivation – if the same event type is derived with only a subset of its attributes then it's a projection type)

The two most common contexts of a rule execution or evaluation are interval (window) and segmentation (group-by) of events:

- The interval context may be a fixed interval where the user specifies the exact times of start and end, a sliding interval where the user specifies how the interval slides either by time or number of events, and an event-based interval where the user specifies the event that will initiate the interval and the event that will terminate it. Only events that occur within the same interval are considered for processing by the agent for that interval, i.e. will satisfy the pattern part of the rule.
- The segmentation context is specified by the user through a data element by which all events are grouped-by if their values match. Only events with the same specified data value will be considered for processing by the agent for a particular segment, i.e. will satisfy the pattern part of the rule.

The syntax of the building blocks is also available for specifying event processing agents programmatically and the artefacts are deployable through APIs to the processing engine (whether single, clustered or distributed). This is for the same reason as to avoid programming sound logical temporal statements.

4.2.2.3 *Critical product attributes*

- Business\Application agility – change patterns rapidly at the end user level, ability to implement and change more rapidly, react and adapt to events, real-time monitoring, continuous intelligence
- Business\Application optimization – early detection, ability to dynamically assemble needed process components at runtime, dynamic services composition and orchestration
- Business\Application efficiency – support decision making, sophisticated action initiation
- Event Processing Network as a CEP logic abstraction towards standardization
- Lower cost of operations – lower maintenance of patterns and rules
- Lower cost of implementation – reduce in design, build and test costs and time
- Scale – increase scale of response and volume, multiple channels of events, widely distributed event sources

4.2.3 Big Data Analysis

4.2.3.1 *Target usage*

Big Data Crunching (also known as Big Data Batch Processing) is the technology used to process huge amounts of previously stored data in order to get relevant **insights** in scenarios where latency is not a highly relevant parameter. These insights take the form of newly-generated data which will be at disposal of applications using the same mechanisms through which initially stored data is available.

On the other hand, Big Data Streaming could be defined as the technology to process continuous unbounded and large streams of data extracting relevant insights on the go. This technology could be applied to scenarios where it is not necessary to store all incoming data or it has to be processed "on the go", immediately after it becomes available. Additionally, this technology would be more suitable to big-data problems where low latency in generation of insights is expected. In this particular case, insights would be continuously generated, parallel to incoming data, allowing continuous estimations and predictions.

Finally, several Real-Time Stream Processing technologies have been defined targeted to real-time generation of insights from a continuous stream of data received at a reasonable input rate.

Lately, a number of commercial solutions for the crunching problem have appeared; most of them based on open-source projects like Hadoop. On the other hand, several Real-Time Stream Processing engines can be found starting from those specialized in particular scenarios, like real-time user modeling for web-advertisement, to those intended to be more generic, like the ones based on Complex Event Processing techniques (see 4.2.2). The approach taken in these two scenarios is radically different, not offering a single elegant solution that can be the most efficient and flexible one for Big Data Crunching scenario while at the same time is able to cope with Big Data Streaming scenarios.

The Big Data Analysis Support GE offers a continuous solution for both Big Data crunching and Big Data Streaming. A key characteristic of this GE is that it would present a unified set of tools and APIs allowing developers to program the analysis on large amount of data and extract relevant insights in both scenarios. Using this API, developers will be able to program Intelligent Services like the ones described in section 4.3. These Intelligent Services will be plugged in the Big Data Analysis GE using a number of tools and APIs that this GE will support.

Input to the Big Data Analysis GE will be provided in two forms: as stored data so that analysis is carried out in batch mode or as a continuous stream of data so that analysis is carried out on-the-fly.

The first is adequate when latency is not a relevant parameter or additional data (not previously collected) is required for the process (i.e. access to auxiliary data on external databases, crawling of external sites, etc). The second is better suited in applications where lower latency is expected.

Algorithms developed using the API provided by the Big Data Analysis GE in oder to process data will be interchangeable between the batch and stream modes of operation. In other words, the API available for programming Intelligent Services will be the same in both modes.

In both cases, the focus of this enabler is in the "big data" consideration, that is, developers will be able to plug "intelligence" to the data-processing (batch or stream) without worrying about the parallelization/distribution or size/scalability of the problem. In the batch processing case, this means that the enabler should be able to scale with the size of the data-set and the complexity of the applied algorithms. On the other hand, in the stream mode, the enabler has to scale with both input rate and the size of the continuous updated analytics (usually called "state"). Note that other GEs in FI-WARE are more focused on real-time response of a continuous stream of events not making emphasis in the big-data consideration (see section 4.2.2).

4.2.3.2 *GE description*

Technologically speaking, big data crunching was revolutionized by Google, introducing a flexible and simple framework called map&reduce. This paradigm allows developers to process big data sets using a really simple API without having to worry about parallelization or distribution. This paradigm is well suited for batch processing in highly distributed data-sets but it is not focused on high-performance and events, so it is less suited for stream-like operations.

The GE we present here is originally based on the map&reduce paradigm but extends it in order to offer high performance in batch mode while still making it suitable for stream processing.

The following diagram offers a general overview of the Big Data Analysis GE, showing main blocks and concepts.

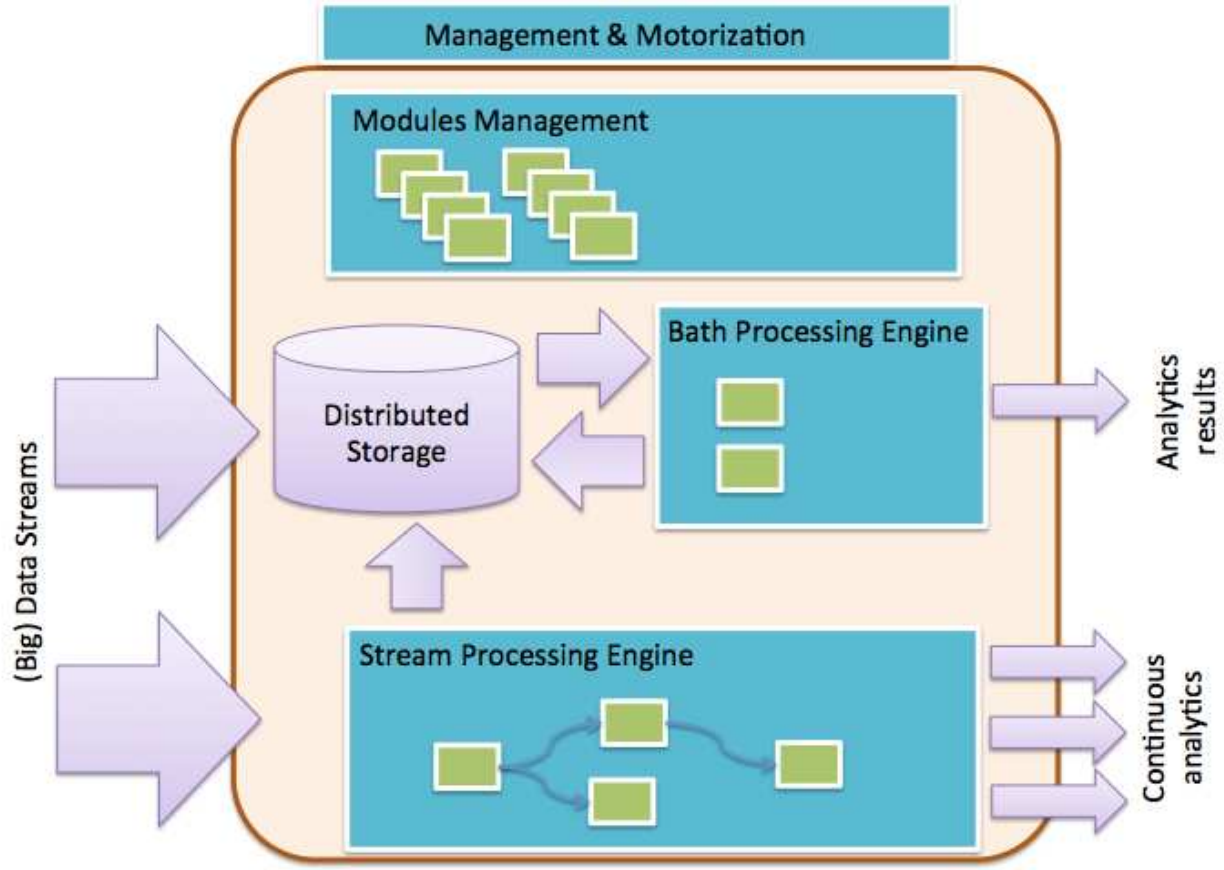


Figure 20 Big Data Analysis GE

First of all, it is important to realize that the module manager contains all algorithms and operations developed to process data and they can be used both in the stream and batch process unit. These modules are developed using a simple API of the enabler inspired by the original map&reduce interface but more focused on intense data processing, allowing the platform to be more oriented to high performance.

The obvious difference between these two engines is that the batch processing engine receives the input data from a distributed storage while the stream engine receives the input data on the fly. Although the execution model is really different, the enabler allows the user to abstract from this difference.

Finally, although the engines (batch and stream) are depicted separately in this figure, they are able to share resources.

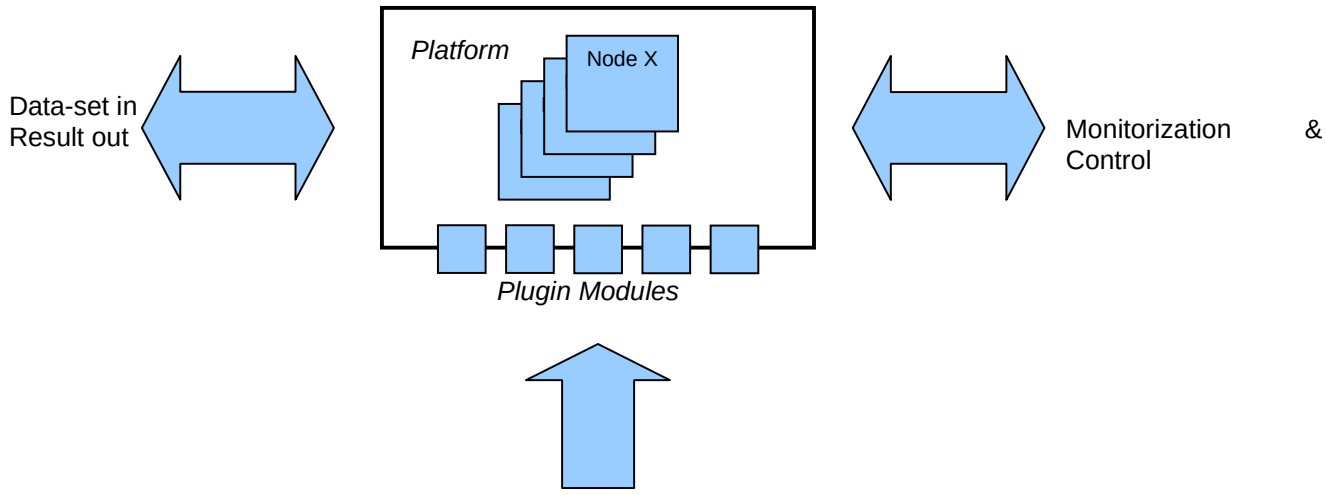


Figure 21 Big-data Platform

What differs this GE from conventional Map-Reduce (MR) platforms are primarily:

- the extensions added to basic MR primitives,
- its stream-oriented MR functionality (a new way to process logs incrementally), and
- its plug-in support system enablng to integrate third party software to process the input data.

This GE is a fully distributed processing engine especially designed for efficient analysis of log-based streams of data.

Prior to execution of this GE, a cluster of nodes needs to be set-up. In this cluster, the data distribution is based on the key-value content, making the platform more efficient since the operations are always based on local data. Also, it allows to implement an efficient stream MR technology. The drawback is when you're facing unevenly distributed data sets.

As all input data is managed by the platform, full control of the process is gained. This increases the efficiency of the platform since all the resources (memory, I/O, CPU cores, etc.) are controlled by the platform itself. This GE would not be a good solution if external data was required (crawling).

We see three mayor areas of benefit using this GE instead of any other known big data platform:

- Efficiency

  This GE is efficient enough to provide low latency analysis for big data sets. Near-real time (or continuous) processing is possible even at a high data input rate.

- Stream Processing

  This GE is able to process continuous input sources updating internal status using stream map&reduce paradigm. This is an important point since Hadoop is too slow to do this, and Yahoo S4 is too light (S4 keeps all the data in memory).

- Big Joins

  “Big Joins” are executed efficiently since this GE has a key-based distribution of all data-sets in the cluster, allowing the execution of joins using local data only. The key point here is not to perform really huge joins (Hadoop can do that), but to perform low-latency join operations.

One of the main ideas behind this GE is to store data locally in the file system of the node where the data is to be processed. This is one of the reasons that this GE is faster, in terms of processing, than any of its competitors.

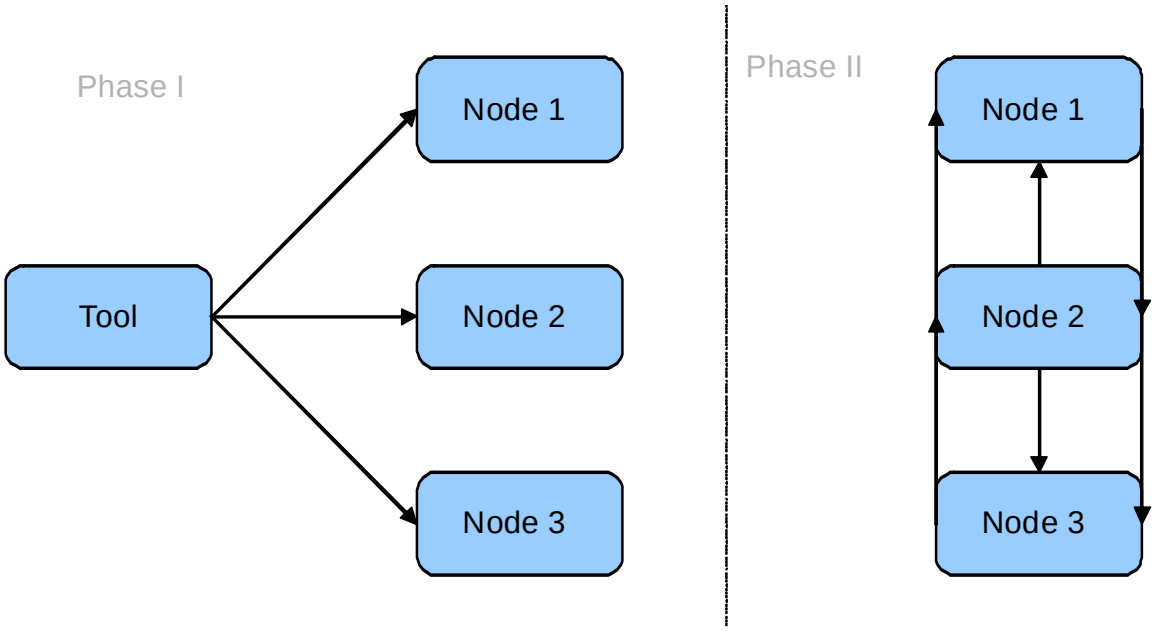


Figure 22 The two phases during upload of data to the platform

When data to be processed is first uploaded to the platform, this data is immediately shuffled and placed on the machine in the cluster where the data is to be processed. This is a huge advantage for the speed of processing, but, it limits the platform in terms of Redundancy and Fault Tolerance of the data residing in the file system. Both Redundancy and Fault tolerance can of course be employed in the platform, but not without implementing them 'from scratch'. We cannot take for example HDFS and use it as the generic distributed file system for the platform; the penalty in processing time would be far too impacting. In fact doing something like that would make the whole concept of this GE loose its meaning. Now, what is planned to be added to the platform is an external storage solution for the final results and the input data. Proprietary Redundancy and Fault tolerance of all the data is in the roadmap.

Execution Module API

Third party libraries to extend the functionality of this GE are added to the platform at run-time. All third party code is executed in separate processes as to not disturb the engine itself. Shared libraries are used for these extensions and the platform is able to add/modify extensions to a running system without pausing or restarting it.

In order to add an execution module for this GE, a third party developer would start by editing a text file for the platform where the whole pluggable functionality is described in a proprietary language. All input and output data types are described here, as well as the number of input sources and output sinks and even scripting code. This input file is used by the platform to create a source code module (C++ is the language adopted in this GE for performance reasons), consisting of a header file and a source code file which the third party developer must edit in order to accomplish the desired operation. A shared library is compiled for each module and these libraries are loaded at runtime by the platform. Serialization of the output data is crucial to the platform, for it to understand how to distribute the data over the cluster, so this part is implemented as platform libraries that the third party developer will use in the development of the module.

For interaction with the platform, a console based tool is developed, connecting the user to the running platform and where he/she can execute his/her modules, pretty much like any shell scripting language.

Key functionality in this scripting language is to:

- upload data sets,

- process these data sets using the extended MR functionality the platform offers (implemented in the module previously described),
- upload the resulting output, and
- supervising the entire platform.

A graphical tool for the same purpose is planned to be part of some future FI-WARE release.

The controlling mechanisms of the platform is in charge of supervising the resources in the cluster and this makes sure the engine is never overloaded, nor that the cluster ever tries to use more memory than what is available. These mechanisms are crucial for an efficient usage of the hardware resources.

4.2.3.3 *Critical product attributes*

- Highly-efficient solution for both batch and stream big-data problems.
- Abstraction of the execution model (batch or stream)
- Flexible interface to develop ad-hoc operations to process incoming streams in both modes
- Low latency continuous complex analytics
- Plug-in functionality, enabling to add 3rd party software in run-time
- Extensions to the traditional set of MR operations enabling more flexible big data analysis

4.2.4 Multimedia analysis

4.2.4.1 *Target usage*

The target users of the Multimedia Analysis GE are all applications that want to extract meaningful information from multimedia content (image, audio, video, etc.) of any kind and that need to automatically find characteristics in multimedia data bases on given tasks and (decision) rules. The GE can work for previously stored multimedia data as well as for multimedia data streams (e.g., received from a camera in real time).

In the media era of the web, much content is user-generated (UGC) and span over any possible kind, from amateur to professional, nature, parties, etc. In such context, multimedia content analysis can provide several advantages for classifying content and later search, or to provide additional information about the content itself.

Today in many situations a photo or and music fragment can be already used to query a system for information, via mobile phone or the web, thus easily enabling the retrieval of information in an real-time and ubiquitous way for the end user.

At the same time, most of such services are vertical silos targeting a single domain or media, in contrast with the general purpose essence of UGC. This calls for brokering systems than can smartly address any type of multimedia content in the best possible way, further leveraging cross-domain knowledge to augment information and recognition itself.

Example applications in different industries addressed by this Generic Enabler are:

- Telecom industry: Identify characteristics in media content (e.g., image/video) recorder by single mobile users; identify communalities in the recordings across several mobile users (e.g., within the same cell).

- Mobile users: (Semi-)automated annotation of recorded content (e.g., images/video), point of interest recognition and tourist information in augmented reality scenarios, social services (e.g., facial recognition).
- IT companies: Automated processing of multimedia content in databases.
- Surveillance industry: Automated detection of relevant events (e.g., alarms, etc.).
- Marketing industry: Object/brand recognition and sales information offered (shops near user, similar products, …).

4.2.4.2 *GE description*

The Multimedia Analysis GE is a modular platform that enables to provide and build services in the field of multimedia recognition easily accessible through a set of APIs for integration in next generation value added services.

Its main characteristics can be summarized as follow:

- It is a single central point of integration of several content detection/recognition technologies, also encompassing real-time processing.
- It is capable of selecting the appropriate processing technologies based on type of content submitted and aggregate responses to provide consolidated results.
- It can further provide additional information based on the recognized content through external Information Providers and/or semantic annotations.
- It provides a central provisioning feature to expand reference datasets for recognition.
- It provides a central feature to provide feedback on the results of the detection/recognition process.
- It offers a set of APIs to allow processing of file and streaming content, both in real-time or on a query base.

Note that not all realizations of the GE necessarily implement all the features, e.g., some do not rely on external Information Providers.

A generic description of the Multimedia Analysis GE is shown in Figure 23. It depicts the generic functional blocks of this GE.

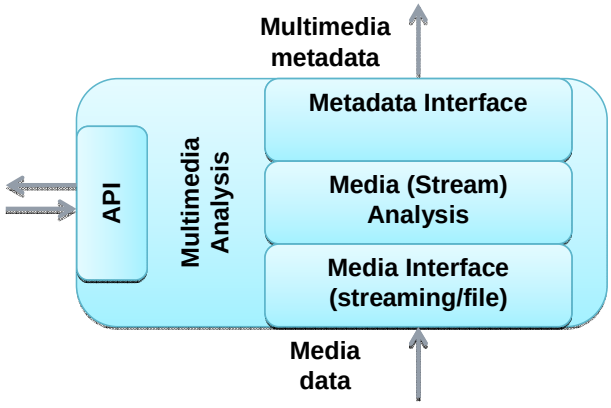


Figure 23 Multimedia Analysis GE – Generic description

The four components of the Multimedia Analysis GE are Media Interface, Media (Stream) Analysis, Metadata Interface, and the API:

- The **Media Interface** receives the media data through different formats. Several streams/files can be accessed in parallel (e.g., different RTP sessions can be handled). Different interchange formats like for streaming and file access can be realized. Example formats are:
  - Real-time Transport Protocol (RTP) as standardized in RFC 3550 [RFC3550]. Payload formats to describe the contained compression format can be further specified (e.g., RFC 3984 [RFC3984] for the H.264/AVC payload).
  - ISO Base Media File Format as standardized in ISO/IEC 14496-12 [ISO 08].
  - HTTP-based interfaces (e.g., REST-like APIs). URLs/URIs could be used to identify the relevant media resources.

  The media type can be, e.g., image content or video content but, in principle, also other types of media content (e.g., audio, speech, etc.) can be processed.
- The **Media (Stream) Analysis** component: As mentioned above, two different types of algorithms can be differentiated for this component: Algorithms operating in the compressed domain and those performing the analysis in the decompressed domain.
  - In the **compressed domain** the media data is analyzed without prior decoding. This allows for low-complexity and therefore resource-efficient processing and analysis of the media stream. The analytics can happen on different semantic layers of the compressed media (e.g., packet layer, symbol layer, etc.). The higher (i.e., more abstract) the layer, the lower the necessary computing power. Some schemes work codec-agnostic (i.e., across a variety of compression/media formats) while other schemes require a specific compression format.
  - Analysis in the **decompressed domain** (e.g., the pixel domain) requires decompression of the media data by a dedicated media decoder prior to the media analysis step.

  Furthermore, different kind of tasks can be addressed by this core component of the GE:
  - Change detection
  - Face detection
  - Object tracking
  - …

  In principle, the analytics operations can be done in real time. In practical implementations, this depends on computational resources, the complexity of the algorithm, and the quality of the implementation. In general, low complexity implementations are targeted for the realization of this GE. In some more sophisticated realizations of this GE (e.g., crawling through a multimedia database), a larger time span of the stream is needed for analytics. In this case, real-time processing is in principle not possible and also not intended.

  The Media (Stream) Analysis component itself can be further structured in sub-components (e.g., for dispatching detectors and recognizers, aggregating results provided by detectors/recognizers, etc. as depicted in Figure 26). These components can also be viewed as part of the API in case they are accessed/configured from outside the GE (see also Figure 24).
- The **Metadata Interface**: A metadata format used for subsequent processing should be used here. The format could, for instance, be HTTP-based (e.g., REST-like APIs) or XML-based. Binary encoding, e.g., using EXI [W3C 11] or BiM [ISO 06], are usually advantages to avoid repeated parsing operations.
- The **API** component is used to access and configure the Multimedia Analysis GE from outside. The different features that can be implemented for this purpose are explained in the following.

Figure 24 describes a more detailed but still generic architecture of the Multimedia Analysis GE. Especially the different APIs on how to access the Generic Enabler are emphasized. Note that in this view, “Image

Recognition Broker" and "Video Processor" are two examples for the Media (Stream) Analysis component, which can optionally inter-work. Since this is a more functional view, the Media Interface and the Metadata Interface are not depicted.

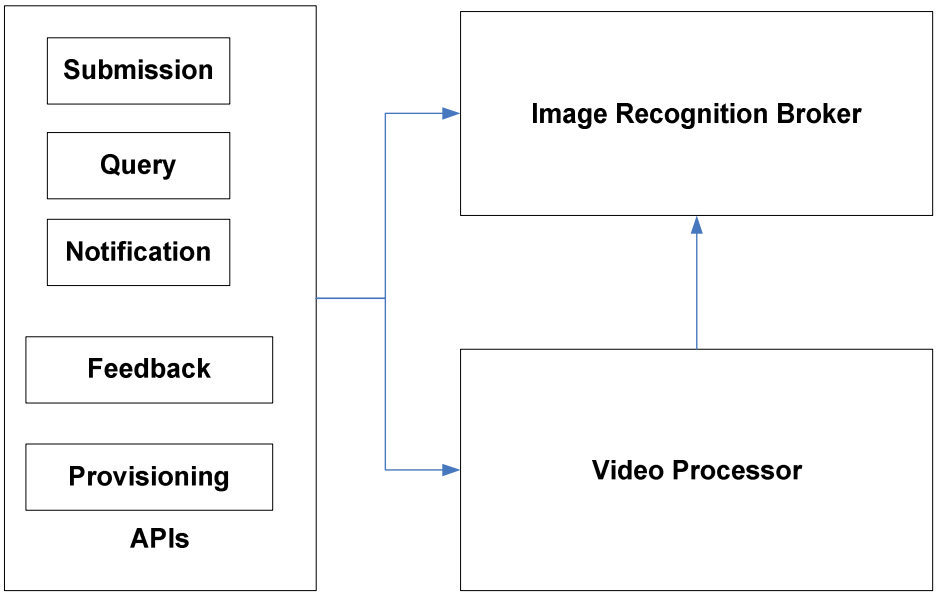


Figure 24 Multimedia Analysis GE – Generic description with detailed interfaces

This enabler relies on three major components:

- APIs
- Image Recognition Broker
- Video processor

The enabler's APIs provide two different ways to reference media when **submitting** the analysis process:

- **Upload mode**: An existing multimedia file (taken from a mobile phone or stored on a server in a file system or in a database) already exists and is used as basic content for processing. The file (e.g., image, video) is uploaded contextually through the API when requesting the analysis process. For analysis, the media file can be accessed independently of the original timing. This means that analysis can happen off-line and random access on the timed media data can be performed. The analysis is performed in the "Media Analysis" operation mode.
- **Reference mode:** The media content is referenced through a URI through the API when requesting the analysis process. This media can either already exist as a file stored on a server or can be a multimedia stream that is generated by a device (e.g., a video camera) and streamed over a network using dedicated transport protocols (e.g., RTP, DASH). In both cases, a reference to this file (e.g., HTTP URL) or stream (e.g., RTSP URL) is provided. When the referenced media is a stream generated in real time, or in case of media playout of a file, the analysis is performed in the "Media Stream Analysis" operation mode.

Besides reference to media, the submission interface also supports the (optional) insertion of:

- metadata such as a title, description, tags, etc, and/or
- callback interface, if notifications are requested.

After submission, the API provides back a reference (identifier) to the submission for matching results. In addition, this API provides different ways to access the results of the analysis (e.g., metadata, augmented information):

- **Query-based interface (pull mode)**: Based on the identifier of the submission, an application can request the status and related metadata/information by polling the enabler.
- **Notification-based interface (push mode)**: In this case, as the analysis process goes on, the enabler notifies the application on the provided callback interface. In case the provided interface contains a HTTP URI, the notification happens periodically. The notification interface can also support streaming URIs to stream notification results if applicable.

The **API** component is in charge of exposing the above functionalities from a functional point of view (e.g., submit a media-based query, poll for metadata/information), but also exposes interfaces for providing feedback to the provided results (and thus improves subsequent results, e.g., confirming the name of the identified face) or to provision datasets for recognition (e.g., to insert a new monument available for recognition). In particular, the **Feedback** and **Provisioning** functionalities are in charge of the internal management of the respective features, further including dispatching requests to the appropriate Processor (image/video) and backend Detectors & Recognizers if applicable, to improve future analyses.

Despite not being decided yet, it is most likely the case that the Multimedia Analysis GE will connect to the Query Broker GE and the Publish/Subscribe Broker GE so that the proposed API will be based on interfaces defined for these two GEs.

Internally to the enabler, the API dispatches submissions to backend processors based on the type of media. Images are sent to the Image Recognition Broker, whilst videos are sent to the Video Processor. Note that in a future release, frames of video streams may be extracted and sent to the Image Recognition Broker to improve the results.

A realization of a Multimedia Analysis GE consists of a composition of different types of realizations for the four building blocks (i.e., components). The core functionality of the realization is determined by the selection of the Media (Stream) Analysis component (and the related sub-components). Input and output format are determined by the selection of the inbound and outbound interface component, i.e., Media Interface and Metadata Interface components. The interfaces can be stream-oriented, but also (for other realizations) can allow polling or pushing of information.

Figure 25 describes two example realizations of the Multimedia Analysis GE and related example usage. Since this example focuses more on the analysis blocks of the enabler, the APIs are not illustrated in this figure.

Two different usage scenarios are regarded:

- **File Access**: A multimedia file has already been generated and is stored on a server in a file system or in a database. For analysis, the media file can be accessed independently of the original timing. This means that analysis can happen slower or faster than real-time and random access on the timed media data can be performed. The analysis is performed in the “Media Analysis” operation mode. Note that in some cases also a streaming interface might be used to realize file access, e.g., in case media playout is realized in addition to (automated) multimedia analysis. In this case, the “Media Stream Analysis” operation mode might be used.
- **Streaming**: A multimedia stream is generated by a device (e.g., a video camera) and streamed over a network using dedicated transport protocols (e.g., RTP, DASH). For analysis, the media stream can be accessed only in its original timing, since the stream is generated in real time. The analysis is performed in the “Media Stream Analysis” operation mode.

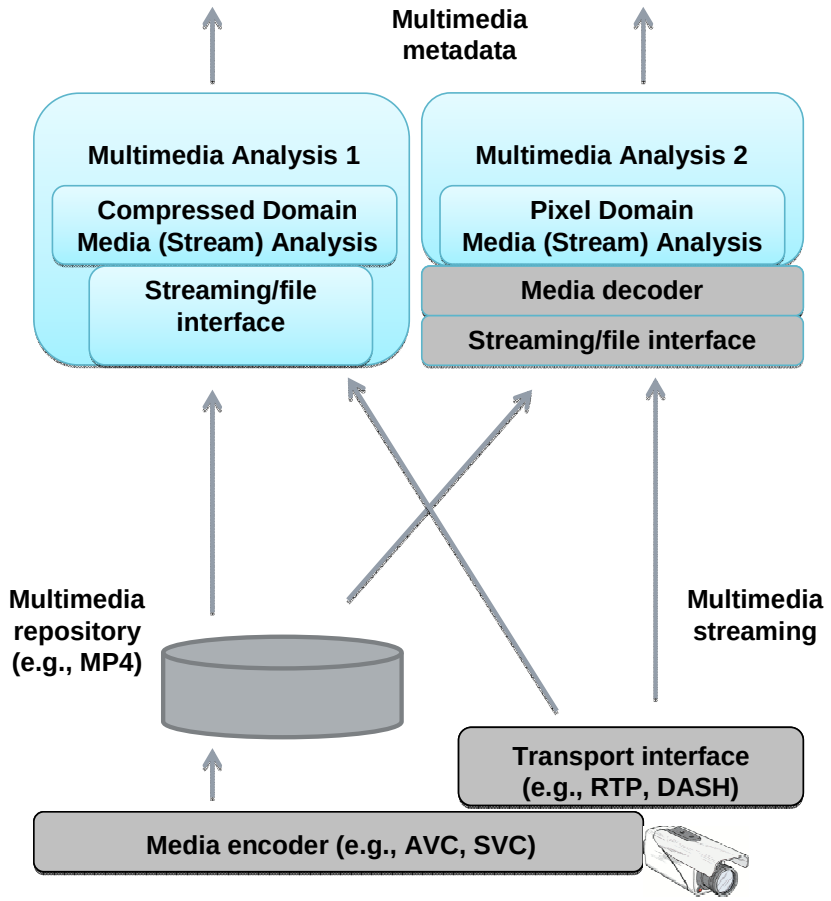


Figure 25 Multimedia Analysis GE – Example realizations

Figure 26 depicts the characteristics of the Image Recognition Broker component as an example for a Media (Stream) Analysis component (gray boxes can be remote).

- The **Dispatcher** component is in charge of selecting the most appropriate set of detectors and/or recognizers to process the content. Possibly, the input information available can also provide metadata to help the analysis process.
- **Detectors** and **Recognizers** are specialized components that relate to specific domains (e.g., music, face, cover, text, …) and/or media types (e.g., audio, image). Such components can be either local (part of the local process) or remote, thus leveraging an external capability of a remote platform. Based on their capabilities, such components can provide results including text, web links, a set of coordinates, a level of confidence, etc. Beside their functional querying interface, such components can provide a provisioning interface (e.g., for managing reference datasets) and/or a feedback interface (e.g., for confirming, refining or invalidating the returned results). The results provided by the Detectors and Recognizers are consolidated within the **Aggregator**, which can further enhance the results with some additional information received from some external providers.
- **Information Providers** are typically remote components used by the aggregator to further enrich the information provided by the Detectors/Recognizers by mashing it with extra content (e.g., based on a remote search, semantic analysis, etc).

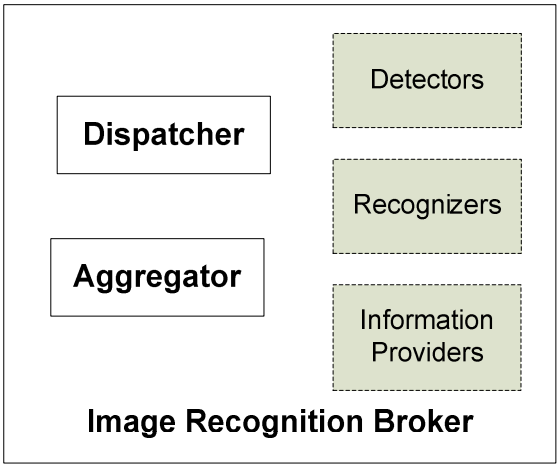


Figure 26 Multimedia Analyser - Image Recognition Broker

Critical product attributes for the Multimedia Analysis GE are especially high detection/recognition ratios containing only few false positives and low-complexity operation. Furthermore, partitioning to independent functional blocks enables the GE to support a variety of analysis methods on several media types and to get easily extended by new features. Even several operations can be combined (e.g., as shown in Figure 24). The mentioned attributes are also reflected in the Critical product attributes listed below.

4.2.4.3 *Critical product attributes*

- General purpose enabler for multimedia analysis and information extraction
- Automated detection of relevant/critical events in media streams
- (Semi-)automated annotation & enrichment of multimedia content
- Pluggable approach for adding specialized detectors & recognizers
- Low complexity algorithms for processing of multimedia data in massively parallel streams and in huge databases
- Parallel processing of a single media stream regarding different criteria
- Fully-featured for feedback and dataset provisioning towards content processors for continuous improvement

4.2.5 Unstructured data analysis

4.2.5.1 *Target usage*

In some domains there is a clear need of using high volumes of unstructured data coming from the Internet (blog posts, rss feeds, new, etc.) in almost real time for a later process and analysis. Target users are any stakeholder that needs to first transform unstructured data from Internet into machine-readable data streams for further almost-real time analysis, decision support systems, etc.

4.2.5.2 *GE Description*

Information is amongst the most valuable assets in the future Internet. Most of the information existing in the current Web is mostly of unstructured nature (blog posts, HTML pages, news, feeds, etc.). The majority of the existing applications today are using only structured data, and therefore overlooking all the potential hidden knowledge that resides in those unstructured resources. There is a clear need of providing a large-scale, near real-time, automatic data acquisition infrastructure that allows the processing of unstructured data from over the Web.

This data acquisition should transform this vast amount of data into processable streams in order to apply the necessary information extraction algorithms needed to transform the raw data into machine-readable information. Algorithms for extraction of high-level features (sentiments, opinions, etc.) tailored for the specific needs of different domains are also needed in this context.

Therefore the system should be able to generate and process massive non-structured and semi-structured data streams in a uniform manner. Once acquired, the data passes through multiple stages where it is processed, resulting in relevant knowledge being extracted. Each stage analyses and processes the received data, enriches it with annotations, and passes it to the next stage. In the final stage, the outcome is presented to the end-user. The whole process can be divided into 4 main stages as illustrated in Figure 27 below.

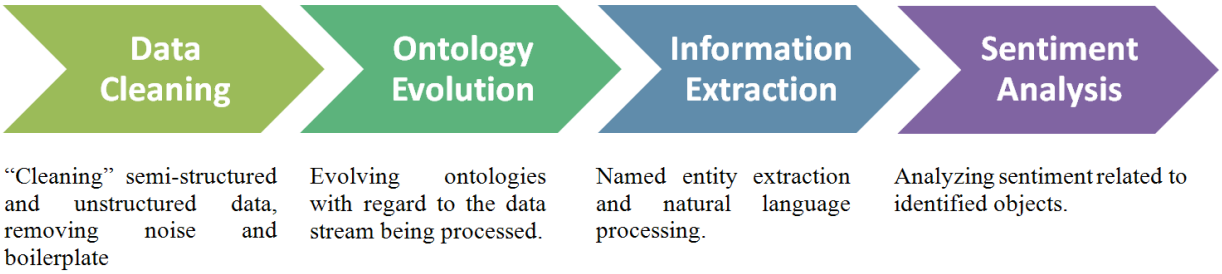


Figure 27 High Level Unstructured data processing enabler

The depicted process covers all functional parts with their proper order and direction of data processing. Pipelining is the fundamental idea of near real-time massive stream processing in the Unstructured Data Processing Enabler. Every stage of the pipeline is able to process data at the same time, ensuring the high throughput that is required for handling massive amounts of data.

The Unstructured Data Processing Enabler pipeline is shown in the figure below.

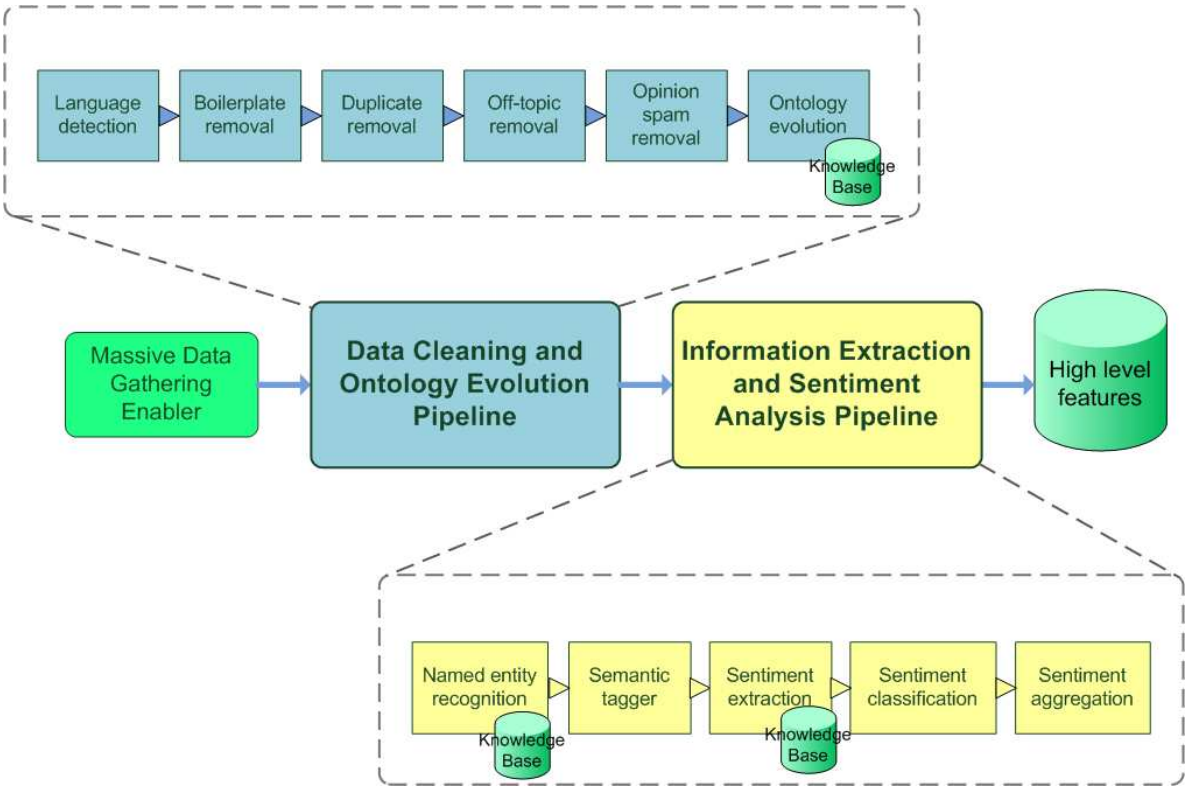


Figure 28 Architecture of Unstructured data processing enable pipeline

From the technical point of view, the pipeline consists of multiple stages (components, depicted as square blocks) that continuously processes data as they are gathered in the form of document streams. First part of the pipeline, called "Data Cleaning and Ontology Evolution" performs necessary preparatory and filtering steps in order to extract only the relevant and raw text from documents streams and make it suitable for later processing. It consists of the following core components:

- Language detection – preliminary filter of documents that cannot be processed because of language limitations. Word-based algorithms or n-gram algorithms play significant role at this step.
- Boilerplate removal – filtering the content of each document, by removing unnecessary and boilerplate information from web documents, such as like advertisements, navigation elements, copyright notices, etc. Numerous techniques might be applied here, such as probabilistic or statistical methods or shallow text features.
- Duplicate removal – ensure that the same or similar documents are not processed twice. To achieve that near-duplicates might be discarded by analysing i.e. each document's simhash.
- Off-topic removal, Opinion spam removal – ensure that only quality data can pass through, thus avoiding spam to bias the computation of high-level features.
- Ontology evolution – semi automatic topic ontology evolution from massive textual stream of documents.

Second part of the processing pipeline "Information Extraction and Sentiment Analysis" is applying NLP techniques in order to extract higher-level features from the document stream, such as sentiments. This second part can be assimilated to an intelligent service generic enabler that extracts sentiments for a particular domain. Other relevant semantic features besides sentiments would potentially be extracted and analysed. In the case of sentiments this enabler offers an ontology-supported process, making use of the knowledgebase from ontology evolution component to ensure improvement of information acquisition over time. Extracted sentiments are stored and aggregated in the way that it is possible to dig down from aggregated sentiment result into the concrete point in the source document.

4.2.5.3 *Critical product attributes*

- Massive web-based information from different sources is extracted, cleaned and transformed to streams.
- The resulting streams are ready to be analysed using different data analysis intelligent services.
- The system provides an ontology evolution module that is capable of evolving an existing ontology for sentiment classification automatically.

4.2.6 Meta-data Pre-processing

4.2.6.1 *Target usage*

Target users are all stakeholders that need to convert metadata formats or need to generate objects (as instantiation of classes) that carry metadata information. The requirements to transform metadata typically stem from the fact that in real life various components implementing different metadata formats need to inter-work. However, typically products from different vendors are plugged together. In this case, the "Metadata Pre-Processor" acts as a mediator between the various products.

4.2.6.2 *GE description*

Figure 29 depicts the components of the "Metadata Pre-Processor" Generic Enabler. These functional blocks are the Metadata Interface for inbound streams, Metadata Transformation, Metadata Filtering, and Metadata/Class Interface for outbound (processed) streams.

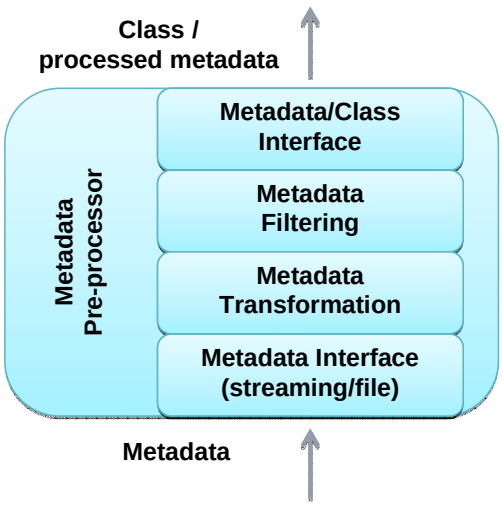


Figure 29 GE "Metadata Pre-Processor"

The functionality of the components is described in the following.

- **Metadata Interface**: This interface for inbound streams. Different interchange formats like for streaming and file access can be realized. An example formats is the Real-time Transport Protocol (RTP) as standardized in RFC 3550 [RFC3550]. Different packetization formats for the contained payload data (i.e., the metadata) depending on the application might be used.

- **Metadata Transformation**: The Metadata Transformation component is the core component of this Generic Enabler. Based on an XML Stylesheet Language for Transformations (XSLT) and a related stylesheet, the processing of the metadata is performed. In principle, also other kind of transforms (other than XSLT) can be applied. The output of this step is an new encapsulation of the metadata received. This could also be a instantiation of a class (e.g., JAVA, C++, C#, etc.)
- **Metadata Filtering**: Metadata Filtering is an optional step in the processing chain. The filtering can be used, e.g., for thinning and aggregation of the metadata, or simple fact generation (i.e., simple reasoning on the transformed metadata).
- **Metadata/Class Interface**: Through this interface, the transformed (and possibly filtered) metadata or metadata stream is accessed. Alternatively, instantiated classes containing the metadata can be received.

Figure 30 shows an example realization of the "Metadata Pre-Processor" Generic Enabler.

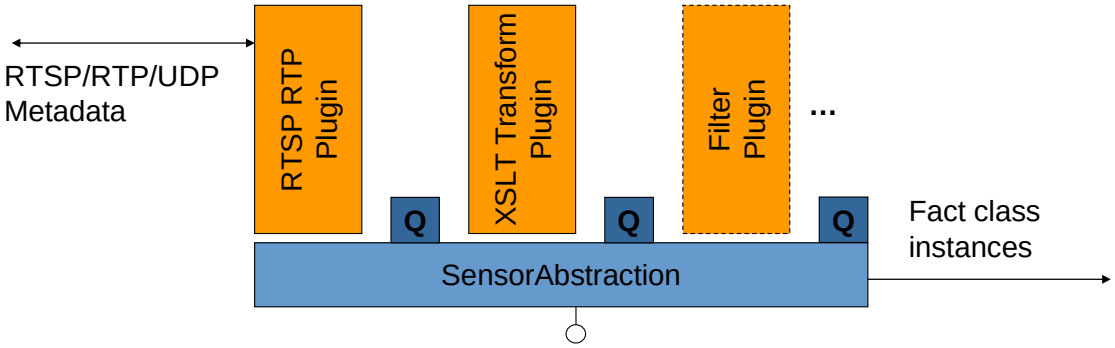


Figure 30 Example metadata pre-processing chain

The internal structure is implemented using a plug-in approach in this example, but this does not need be the case necessarily. In the example, timed metadata is received over an RTSP/RTP interface, which implements the metadata interface for inbound data/streams. Different RTP sessions can be handled; therefore metadata streams can be received from several devices (e.g., cameras or other type of sensors). The target in such a realization could be the provision of metadata as facts to the metadata broker, which would be the receiver of the outbound stream. Internally the metadata items ('facts') are represented by one class per task, which leads to Java classes with flat hierarchy. (In principle, also derivation of classes for the representation of metadata could be used if there are advantages for the application.) As mentioned above, the Metadata Transformation itself is performed by an XSLT stylesheet. The schema for this transform defines a XML-to-'XML serialized JavaBeans' transformation, which produces the Java classes that incorporate the metadata. Individual stylesheets can be used in order to customize the different metadata schemas. Further metadata filtering can be plugged in, which, however, was not necessary in the imaged application of XML-to-'XML serialized JavaBeans' transformation.

The external API is yet to be defined, but we envision a RESTful API that permits easy integration into web services other components requiring metadata access and transformation services.

4.2.6.3 *Critical product attributes*

- Encapsulation of transport and metadata transformation as-a-service, usable from other web applications or components
- Generic metadata transformation approach
- Transformation based on standardized and commonly used XML Stylesheet Language for Transformations (XSLT).

- In addition to encapsulation in (XML- or JSON-based) metadata formats, also incorporation of the metadata into objects (e.g., serialized Java/C++/C# classes) can be realized (by simply exchanging the stylesheet for the XSLT).

4.2.7 Localization Platform

4.2.7.1 *Target usage*

The Localization GE in FI-WARE targets any application, GEs in FI-WARE, or any complementary platform enabler, that aims to retrieve mobile device positions and localization area events. The localization GE is based on various positioning techniques such as A-GPS, WiFi and Cell-Id whilst taking into account the end-user privacy.

This GE addresses issues related to localization of mobile devices in difficult environments such as urban canyons and light indoor environments where the GPS receiver in the mobile device is not able to acquire GPS signals. It improves GPS coverage whilst waiting for a GPS position, which helps to enhance the user experience of end-users using location-aware applications through their mobile handsets, and the performance of applications requesting the position of mobile devices.

4.2.7.2 *GE description*

The following figure describes the main modules of the Localization GE, also called SUPL Location Platform (SLP) as detailed in the Open Mobile Alliance (OMA) standard. This platform relies on two fully standardised protocols:

- **SUPL**: The "Secure User Plane Location" protocol facilitates the communication between the SUPL Location Platform (SLP) and any SUPL Enabled Terminal (SET) over TCP/IP. Two main scenarios are standardised:
  - Set-initiated: the SET requests GPS assistance data from the SLP to compute a GPS fix or requests the computation of a WiFi position from measurements,
  - Net-initiated: a third party application requests the position of a SET via MLP (see below) which triggers the sending of a binary SMS to the SET by the SLP. The mobile can then communicate with the SLP over TCP/IP to exchange assistance data, WiFi measurements or send event reports.
- **MLP**: The "Mobile Location Protocol" facilitates the communication to and from an application such as Yellow Pages over HTTP/XML. The request contains various parameters that are used by the SLP to determine the best or preferred location method to use and return mobile positions or event reports. Another alternatives for the interface between the SLP and the applications, wrapping usage of MLP, are being considered but are still under discussion (see section 4.4.2).

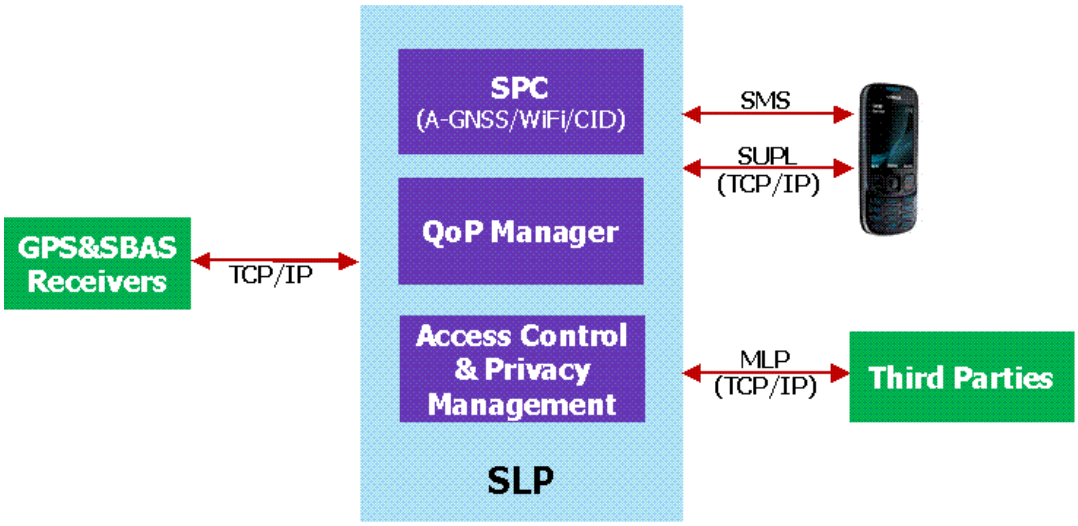


Figure 31 SUPL Location Platform

The following describes the main modules of the SLP:

- **Access Control and Privacy Management**: A third party requesting the location of an end-user using a SET is first authorised based on login/password, application invoked, number of location requests per month, per second. If the request is accepted, the privacy of the end-user is then verified based on a global profile for all applications or based on a specific application profile. The end-user configures its profiles via SMS or web-pages (self-care); a few examples are listed below:
  - Localisation allowed permanently, once, on specific time windows of the day or refused,
  - Localisation allowed for specific level of accuracy: low (CID), medium (WiFi), high (GPS),
  - Localisation allowed for list of friends at the origin of the location request (contained in MLP request), being the list of friends managed by the proper Security, Trust and Privacy GEs
  - Option to receive notification SMS each time the end-user is being localised,
  - Option to active or deactivate the caching of its location.
- **Quality of Positioning (QoP) Manager**: Based on parameters contained in MLP request or set-init request and the rough location of the SET (cell-id), the SLP is able to select the best positioning method to use. This selection mechanism is dynamically configurable in the internal cell database and multiple location technologies can be triggered.

  The criteria for the location technology support in the terminal can also be fully configured inside the cell database. For example, yellow pages may want a very quick and rough location, where a “find a friend” application may be more permissive regarding latency in getting a friend location.

  When multiple locations are returned by the terminal, the QoP manager is in charge of selecting the best location to use or even perform hybridisation of those locations to generate the final position fix.

  A coherence check of the two locations is also performed to ensure the integrity of the end-user location; this feature is also called location authentication.
- **SPC**: The SUPL Positioning Centre has in charge the position calculation of the SET, based on the following positioning techniques:
  - **A-GPS**: Based on GPS&SBAS receivers, the SPC computes assistance GPS data that is used by the SET to enhance its time to first fix and receiver sensitivity, and offer a worldwide coverage of the service. The platform offers additional enhancements related to GPS integrity (allowing the detection of a faulty GPS satellite) and GPS differential corrections used to smooth pseudo-distance measurement degradations. This technique requires the SPC to know the rough location of the

terminal with an uncertainty of hundreds of kilometres. The Cell Id data base is either provided by third party, or can be automatically provisioned.

- **WiFi**: Based on WiFi hotspot signal strength measurements sent by the SET to the SLP, the SPC is able to compute a position by standard triangulation technique. This technique requires the SPC to have a direct mapping between hotspot Medium Access Control (MAC) address and its position.
- **Cell-Id**: Cell identifiers sent by the terminal to the SPC are converted to a position thanks to the internal cell database, which can be provisioned dynamically.

The SUPL Positioning Centre has also in charge the event triggering and processing of event reports from the terminal as requested by a third-party via the MLP interface. The event triggering facilitates the following scenarios:

- **Inside**: Each time the terminal is within a specific area, it will send a report back to the SPC which is transferred back to the original third-party application as an event-driven response (Trigger Location Report). Note that it is possible to command the terminal to send periodic reports at configurable intervals as long as it is within the area.
- **Outside**: Each time the terminal is outside a specific area, a report will be sent back to the third-party application,
- **Entering**: Each time the terminal enters a specific area, a report will be sent back to the third-party application,
- **Leaving**: Each time the terminal leaves a specific area, a report will be sent back to the third-party application.

All those reports are based on the area requested by the third-party application in the MLP request, which can be a polygon, a list of GSM cells or even one single cell. In case a polygon is requested, the SPC computes all GSM cells that are within this polygon and borderline to make the terminal computation easier.

Event triggering is illustrated on the following figure, taken from OMA SUPL standard with a third-party application requesting periodic reporting of mobile inside a specific cell:

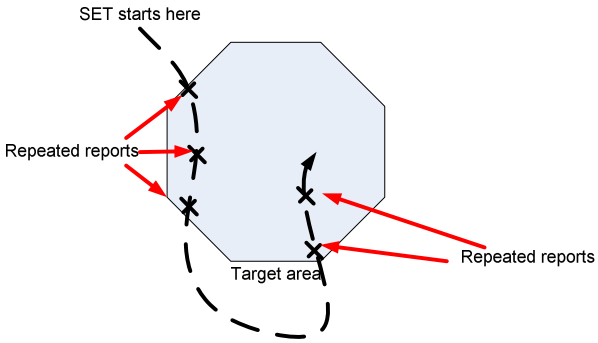


Figure 32 Event Triggers

4.2.7.3 *Critical product attributes*

- Provides mobile location and geo-fencing events based on standard lightweight protocols.
- Fully configurable end-user privacy management per third-party application.
- Best in class GPS assistance data allowing a high service availability, worldwide.
- Dynamic location technology selection based on end-user environment.

4.2.8 Query Broker

4.2.8.1 *Target usage*

The Query Broker GE provides an intelligent, abstracting interface for retrieval of data from the FI-WARE data management layer. This is provided in addition to the publish/subscribe interface as another modality for accessing data.

Principal users of the Query Broker GE include applications that require a selective, on-demand view on the content/context data in the FI-WARE data management platform via a single, unified API, without taking care about the specifics of the internal data storage and DB implementations and interfaces.

Therefore, this GE provides support for integration of query-functions into the users' applications by abstracting the access to databases and search engines available in the FI-WARE data management platform while also offering the option to simultaneously access outside data sources. At the same time its API offers an abstraction from the distributed and heterogeneous nature of the underlying storage, retrieval and DB / metadata schema implementations.

The Query Broker GE provides support for highly regular ("structured") data such as the one used in relational databases and queried by SQL like languages. On the other hand it also supports less regular "semi-structured" data, which are quite common in the XML tree-structured world and can be accessed by the XQuery language. Another data structure supported by the Query Broker is RDF as a well structured graph-based data model that is queried using the SPARQL language. In addition, the Query Broker GE provides support for specific search and query functions required in (metadata based) multimedia content search (e.g., image similarity search using feature descriptors).

The question about how non-relational or "NoSQL" databases, which are becoming an increasingly important part of the database landscape, can be integrated, is one of the open points to be addressed during the FI-WARE project.

The underlying approach for the extension of the Query Broker GE is to try identifying families of (abstract) query languages (based on minimum common denominators of existing query languages) together with preferred representatives allowing to categorize the capabilities of the data resources in respect to what and how they can be queried.

4.2.8.2 *GE description*

Main Functionality

The Query Broker GE is implemented as a middleware to establish unified retrieval in distributed and heterogeneous environments with extensions for integrating (meta-)data in the query and retrieval processes. To ensure interoperability between the query applications and the registered database services, the QueryBroker is based on the following internal design principles:

- Query language abstraction:

  The Query Broker GE will be capable to handle queries formulated in any of a defined set of query languages/APIs (e.g., XQuery or SQL or SPARQL) used by the services for retrieval. Following this concept, all incoming queries will be converted into an internal abstract format that will be then translated into the respective specific query languages/APIs when accessing the actual data repositories. Addressing this requirement, this abstract query format may be based on and extend XQuery functionalities. As an example, this format may be based on the MPEG Query Format (MPQF) [Smith 08], which supports most of the functions in traditional query lnaguages and also incorporates several types of multimedia specific queries (e.g., temporal, spatial, or query-by-example). By this, requests focusing on data centric evaluation (e.g., exact matches by comparison operators) are inherently supported.

- Multiple retrieval paradigms

  Retrieval systems are not always following the same data retrieval paradigms. Here, a broad variety exists, e.g. relational, No-SQL or XML-based storage or triple stores. The Query Broker GE attempts to shield the applications from this variety. Further, it is most likely in such systems, that more than one data base has to be accessed for query evaluation. In this case, the query has to be segmented and distributed to applicable retrieval services. This way, the Query Broker GE acts as a federated database management system.

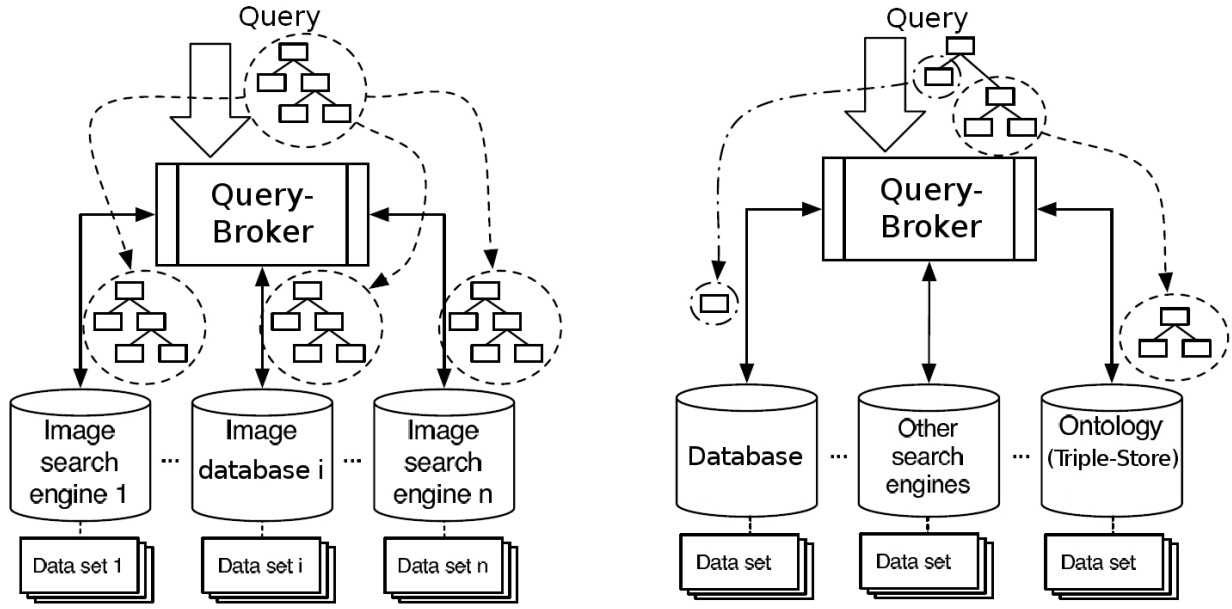
- Metadata format interoperability:

  For an efficient retrieval process, metadata formats are used to describe syntactic or semantic attributes of resources. Currently there exist a huge number of standardized/proprietary metadata formats for nearly any use case or domain. Therefore it can be estimated, that more than one metadata format is in use in a heterogeneous retrieval scenario. The Query Broker GE therefore provides functionalities to perform the transformation between diverse metadata formats where a defined mapping exists and is made available.

Query Processing Strategies

The Query Broker GE is a middleware that can be operated in different facets within a distributed and heterogeneous search and retrieval framework including multimedia retrieval systems. In general, the tasks of each internal component of the Query Broker (see Figure 34) depend on the registered databases and on the use cases.

In this context, two main query processing strategies are supported, as illustrated in Figure 33



(a) Local processing **(b) Distributed processing**

Figure 33 Query processing strategies

The first paradigm deals with registered and participating retrieval systems that are able to process the whole query locally, see Figure 33(a). In this sense, those heterogeneous systems may provide their local metadata format and a local / autonomous data set. A query transmitted to such systems is understood as a whole and the items of the result set are the outcome of an execution of the query. In case of differing metadata formats in the back ends a transformation of the metadata format may be needed before the

(sub)query is transmitted. In addition, depending on the degree of overlap among the data sets, the individual result sets may contain duplicates. However, a result aggregation process only needs to perform an overall ranking of the result items of the involved retrieval systems. Here, duplication elimination algorithms may be applied as well.

The second paradigm deals with registered and participating retrieval systems that allow distributed processing on the basis of a global data set, see Figure 33 (b). The involved heterogeneous systems may depend on different data representation (e.g., ontology based semantic annotations and XML-based feature values) and query interfaces (e.g., SPARQL and XQuery) but describe a common (linked) global data set. In this context, a query transmitted to the QueryBroker needs to be evaluated and optimized, which results into a specific query execution plan. In series, segments of the query are forwarded to the respective engines and executed. Now, the result aggregation has to deal with a correct consolidation and (if required) format conversion of the partial result sets. In this context, the Query Broker GE behaves like a federated Database Management System.

QueryBroker Architecture

Figure 34 illustrates an end-to-end workflow scenario in a distributed retrieval scenario. At its core, the Query Broker GE transforms incoming user queries (of different formats) to a common internal repesentation for further processing and distribution to registered data resources and aggregates the returned results before delivering it back to the client. In the following, the subcomponents of a potential reference implementation of the Query Broker GE, based on internal usage of the the MPEG Query Format (MPQF), are briefly described.

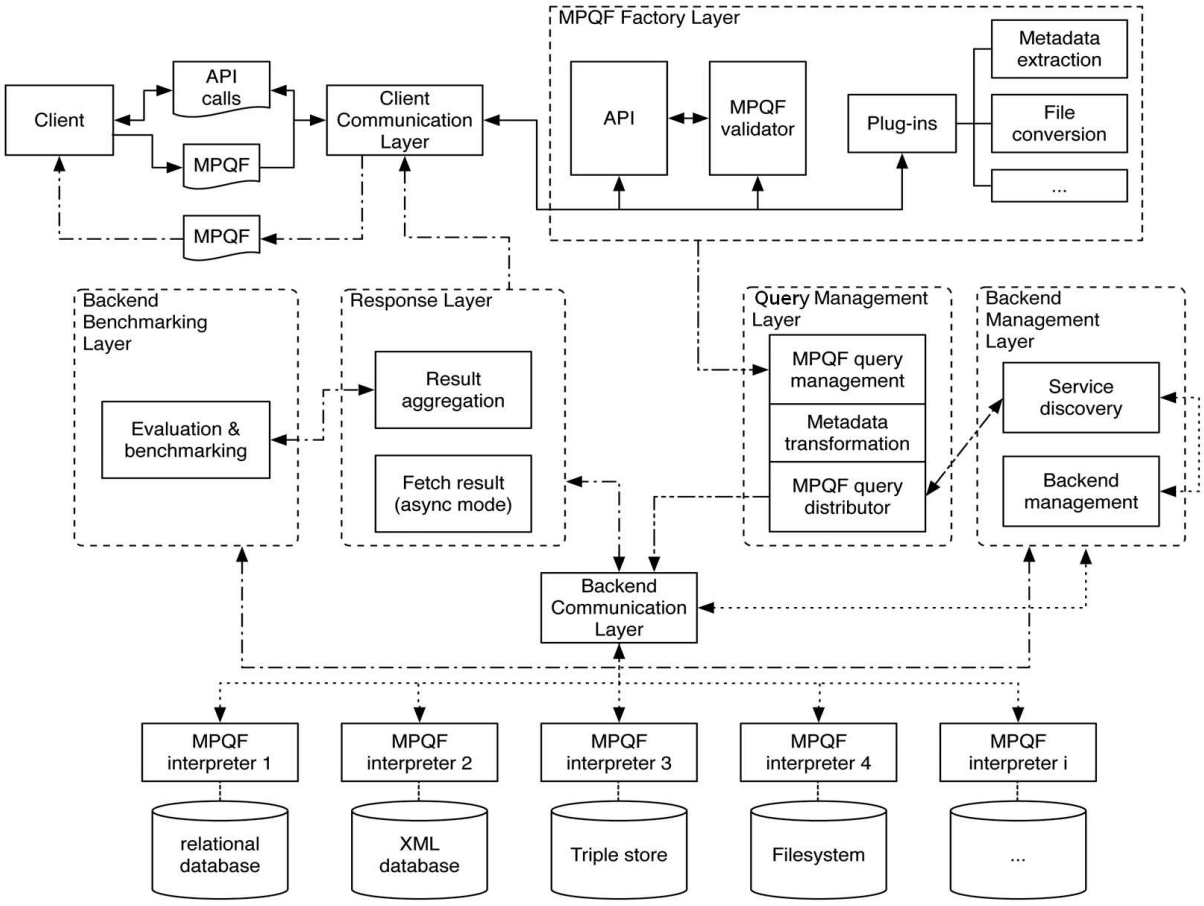


Figure 34 Architecture of the QueryBroker

Backend Management Layer

The main functionalities of the Backend Management Layer are the (de-)registration of backends with their capability descriptions and the service discovery for the distribution of queries. These capability descriptions are standardized in ISO 15938-12, allowing the specification of the retrieval characteristics of registered backends. Such characteristics consider for instance the supported query types or metadata formats. In series, depending on those capabilities, this component is able to filter registered backends during the search process (service discovery). For a registered retrieval system, it is very likely that not all functions specified in the incoming queries are supported. In such an environment, one of the important tasks for a client is to identify the backends which provide the desired query functions or support the desired result representation formats identified by e.g. an MIME type using the service discovery.

MPQF Factory Layer

The main purpose of the MPQF Factory Layer is the generation and validation of (internal) MPQF queries. The transformation of incoming user queries is handled through an API. In general, the internal MPQF query representation consists of two main parts. First, the QueryCondition element holds the filter criteria in an arbitrary complex condition tree. Second, the OutputDescription element defines the structure of the result set. In this object, the needed information about required result items, grouping or sorting is stored. After finalizing the query creation step, the generated MPQF query will be registered to the QueryBroker. A set of query templates at the client side can be established to simplify the query creation process using the API approach. In case an instance of a query is created at the client side in MPQF format then this query will be directly registered to the QueryBroker.

This layer optionally also encapsulates interfaces for inserting preprocessing plug-ins. These could for example support perform file conversations.

Query Management Layer

The Query Management Layer organizes the registration of queries and their distribution to the applicable retrieval services. After the registration with a unique identifier of the entire query, the distribution of the query depends on the underlying search concept. For the local processing scenario, the whole query is transmitted to the backends in parallel. In contrast to that, in a distributed processing scenario, the query will be automatically divided in segments by analyzing the query types used in the condition tree. Here, well known tree algorithms like depth-first search can be used. The key intention of this segmentation is that every backend only gets a query segment, which it can process as a whole. In addition, the transformation between metadata formats is another task of the management layer. In order to monitor and manage the progress of received queries, the QueryBroker implements the following query lifecycle: pending (query registered, process not started), retrieval (search started, some results missing), processing (all results available, aggregation in progress), finished (result can be fetched) and closed (result fetched or query lifetime expired). These states are also valid for the individual query segments, since they are also valid MPQF queries.

MPQF Interpreter

MPQF interpreters act as a mediator between the QueryBroker and a particular retrieval service. An interpreter receives an MPQF formatted query and transforms it into native calls of the underlying query language of the backend database or search engine system. In this context, several interpreters (mappers) for heterogeneous data stores have been implemented (e.g., Flickr, XQuery, etc.). Furthermore, an interpreter for object- or relational data stores is envisaged. After a successful retrieval, the Interpreter converts the result set in a valid MPQF formatted response and forwards it to the QueryBroker.

Response Layer and (planned) Backend Benchmarking Layer

The Response Layer performs the result aggregation and returns the aggregated result set. The current implementation provides a Round Robin aggregation mechanism. Additional result aggregation algorithms are under consideration [Döller 08b], which also could take advantage of the Backend Benchmarking Layer.

In order to describe the main advantage of the (planned) Backend Benchmarking Layer (BBL), let us assume a scenario as shown in Figure 33(a). There, for instance image retrieval may be realized by a query by example search. A query may be sent directly to the Query Broker GE and the whole query is distributed to the applicable backends. The major task in this case is not the distribution, but the result aggregation of the different result sets. The Query Broker GE has to aggregate the results on the one side by eliminating all duplicates and on the other side by performing a ranking of the individual result items. The initial implementation uses the round robin approach which provides efficient processing of result sets of autonomous retrieval systems. However, it is supposable that different backends use different implementations and quality measures for processing the fuzzy retrieval leading to quality discrepancies between the result sets. Therefore, similar to approaches such as [Crashwell 99] where statistics about sources are collected, the BBL will provide information about the search quality of a supporting a more intelligent re-ranking and aggregation of the result set. This information and a respective query classification model may be realized by a new benchmarking environment that allows to rate the search quality of registered backends. This subcomponent is currently under investigation.

4.2.8.3 *Critical product attributes*

- Middleware component for unified access to distributed and heterogeneous repositories (with extensions supporting multimedia repositories)
- Provisioning of metadata format interoperability via schema transformation
- Abstraction from heterogeneous retrieval paradigms in the underlying data bases and search engines

4.2.9 Semantic Annotation

4.2.9.1 *Target usage*

Target users are all stakeholders that want to enrich textual data (tags or text) with meaningful and external content.

In the media era of the web, much content is text-based or partially contains text, either as media itself or as metadata (e.g. title, description, tags, etc). Such text is typically used for searching and classifying content, either through folksonomies (tag-based search), predefined categories, or through full-text based queries. To limit information overload with meaningless results there is a clear need to assist this searching process with semantic knowledge, thus helping in clarifying the intention of the user. This knowledge can be further exploited not only to provide the requested content, but also to enrich results with additional , yet meaningful content, which can further satisfy the user needs.

Semantics, and in particular Linked Open Data (LOD), is helpful in both annotating & categorizing content, but also in providing additional rich information that can improve the user experience.

As end-user content can be of any type, and in any language, such enabler requires a general purpose & multilingual approach in addressing the annotation task.

Typical users or applications can be thus found in the area of eTourism or eReading, where content can benefit from such functionality when visiting a place or reading a book, for example being provided with additional information regarding the location or cited characters.

The pure semantic annotation capabilities can be regarded as helpful for editors to categorize content in a meaningful manner thus limiting ambiguous search results (e.g. an article wouldn't be simply tagged with apple, but with its exact concept, i.e. a fruit, New York City or the brand)

4.2.9.2 *Description of Generic Enabler*

Figure 29 depicts the components of the "Semantic Annotator" Generic Enabler. The internal functional blocks are the Text Processor, the Semantic Broker and related resolvers, the Semantic Filter, the Aggregator, and the API.

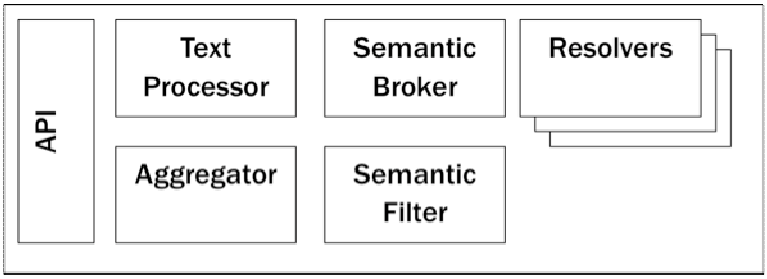


Figure 35 GE "Semantic Annotation"

The functionalities of the components are described in the following.

- **Text Processor**: this component is in charge of performing a first analysis of the text to be annotated, namely language detection, text cleaning, and natural-language processing. The goal of this component is to pre-process the original text to extract useful information for the next step, for example by identifying specific terms that may be of higher interest for the user. Depending on the tools used in this component, a rich multiword Named Entity Recognition can be performed, as well as some generic text classification to assist the following process.
- **Semantic Broker**: this component is composed of a broker itself, assisted by a set of resolvers that can perform full-text or term-based analysis based on the previous output. Such resolvers are aimed at providing candidate semantic concepts referring to Linked Open Data as well as additional related information if available. The exact set of resolvers, and how they are invoked is an implementation issue. Resolvers may be domain- or language-specific, or general purpose.
- **Semantic Filter**: Such component is essential in filtering out candidate LOD concepts coming from the broker, and can include algorithms for scoring results (potentially provided by the single resolvers), ranking & validating candidates. This component is thus responsible for solving disambiguations by comparing results together, and with the original context to achieve the best fitted concept. This filter can further cross-check consistency & relation between candidate concepts to improve disambiguation.
- **Aggregator**: This component is in charge of further expanding information related to the concepts identified after the filtering process, thus possibly aggregating information from related concepts, and/or from several resolvers to provide a unique composite view of the concept related to a textual term.
- **API**: Through this interface, the semantic annotation process is triggered and provides the candidate LOD concepts and related links.

Figure 36 shows an example realization of the "Semantic Annotator" Generic Enabler.

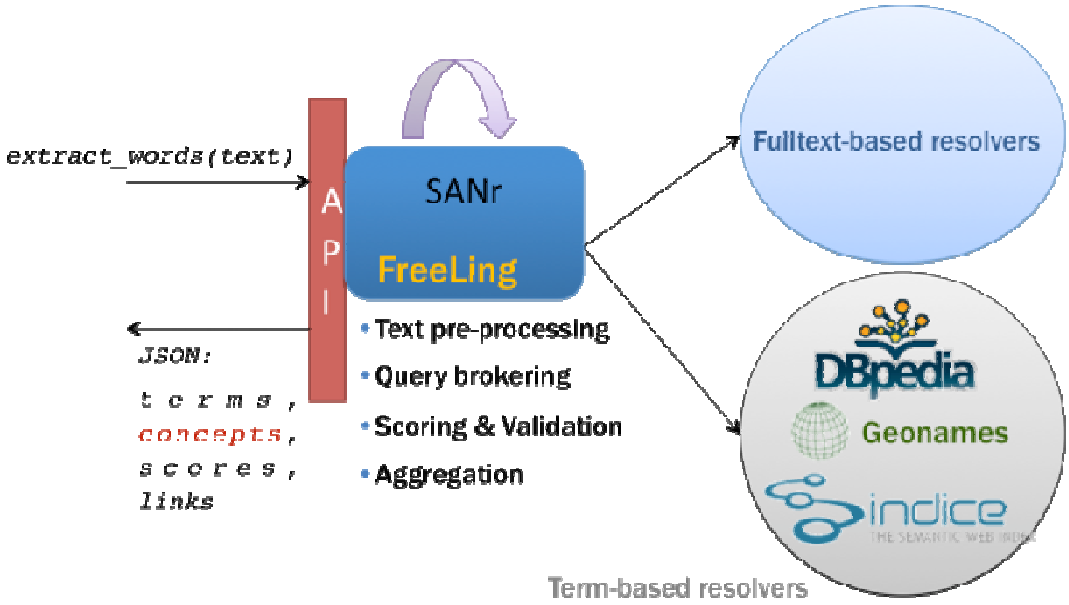


Figure 36 Example Semantic Annotator

The internal structure of this example implementation uses a plug-in approach for invoking resolvers, but this does not need be the case necessarily. In the example, the Text processor is implemented using the FreeLing language analyzer, coupled with a language analyzer to identify the text language first. This feature distinguish the enabler from most current annotators (e.g. Zemanta, Evri) which are mostly focused on English-based concepts and content.

The resolvers used in this implementation are mainly DBPedia and Geonames (for location concepts only) as they proof to be generic enough to annotate any type of text.

The current implemented filter is applying an algorithm based on syntactic and semantic matching of the concepts against the original terms.

The API module provides a JSON-based interface to remotely interact with the enabler. Such API provides a list of candidate concepts and related information for each term considered 'relevant' within the original text.

4.2.9.3 *Critical product attributes*

- Semantic annotation as a service: general purpose enabler to provide related LOD concepts and information to any text
- Dual usage: editorial (semantic classification) and end-user (content augmentation)
- Multilingual support
- Pluggable approach for adding resolvers

4.2.10 Semantic Application Support

4.2.10.1 *Target usage*

Target users are mainly ontology engineers and developers of semantically-enabled applications that need RDF storage and retrieval capabilities. Other GE from the FI-WARE, such as for example the GE for semantic service composition or the query broker, or from the usage areas of the PPP that need semantic infrastructure for storage and querying are also target users of this GE.

4.2.10.2 *Description of GE*

Ten years had passed since Tim Berners-Lee envisioned a new future for the Web, the Semantic Web. In this future, the Web, that had been mostly understandable by humans, will evolve into a machine understandable Web, increasing its exploitation capabilities. During these years, Semantic Web has focused the efforts of many researchers, institutions and IT practitioners. As a result of these efforts, a large amount of mark-up languages, techniques and applications, ranging from semantic search engines to query answering system, have been developed. Nevertheless the adoption of Semantic Web from IT industry has been a hard and slow way.

In the past few years, several authors had theorized about the reasons preventing Semantic Web paradigm adoption. Reasons can be categorized into two main subjects: technical reasons and engineering reasons. Technical reasons focus on the lack of infrastructure to meet industry requirements in terms of scalability, distribution, security, etc. Engineering reasons stress that methodologies, best practices and supporting tools are needed to allow enterprises developing Semantic Web applications in an efficient way.

Semantic Application Support enabler will address both engineering and technical aspects, from a data management point of view, by providing:

- An infrastructure for metadata publishing, retrieving and subscription that meets industry requirements like scalability, distribution and security.
- A set of tools for infrastructure and data management, supporting most adopted methodologies and best practices.

Therefore Semantic Web Application Support enabler will allow users of the GE efficient and effective development of high quality Semantic Web based applications.

Figure 37 illustrates the architecture of the Semantic Web Application Support Enabler.

Three main areas can be identified: Semantic Infrastructure, Semantic Engineering and External Components.

- Semantic Infrastructure contains services and APIs providing core functionalities for both Semantic Engineering and External components.
- Semantic Engineering contains tools and services built on top of Semantic Infrastructure functionality that provides ontology management and engineering features to human users.
- External Components are the clients of the functionality provided by this GE. It contains software agents that take advantage of the functionality provided by the Semantic Infrastructure. In the scope of FI-WARE, two main kinds of External Components can be foreseen: GE from other FI-WARE areas and applications developed by FI-WARE Usage Areas. As Semantic Engineering and External Components share the same Semantic Infrastructure, humans can use engineering functionality to easily modify and mange the behaviour of External Components by modifying stored information.

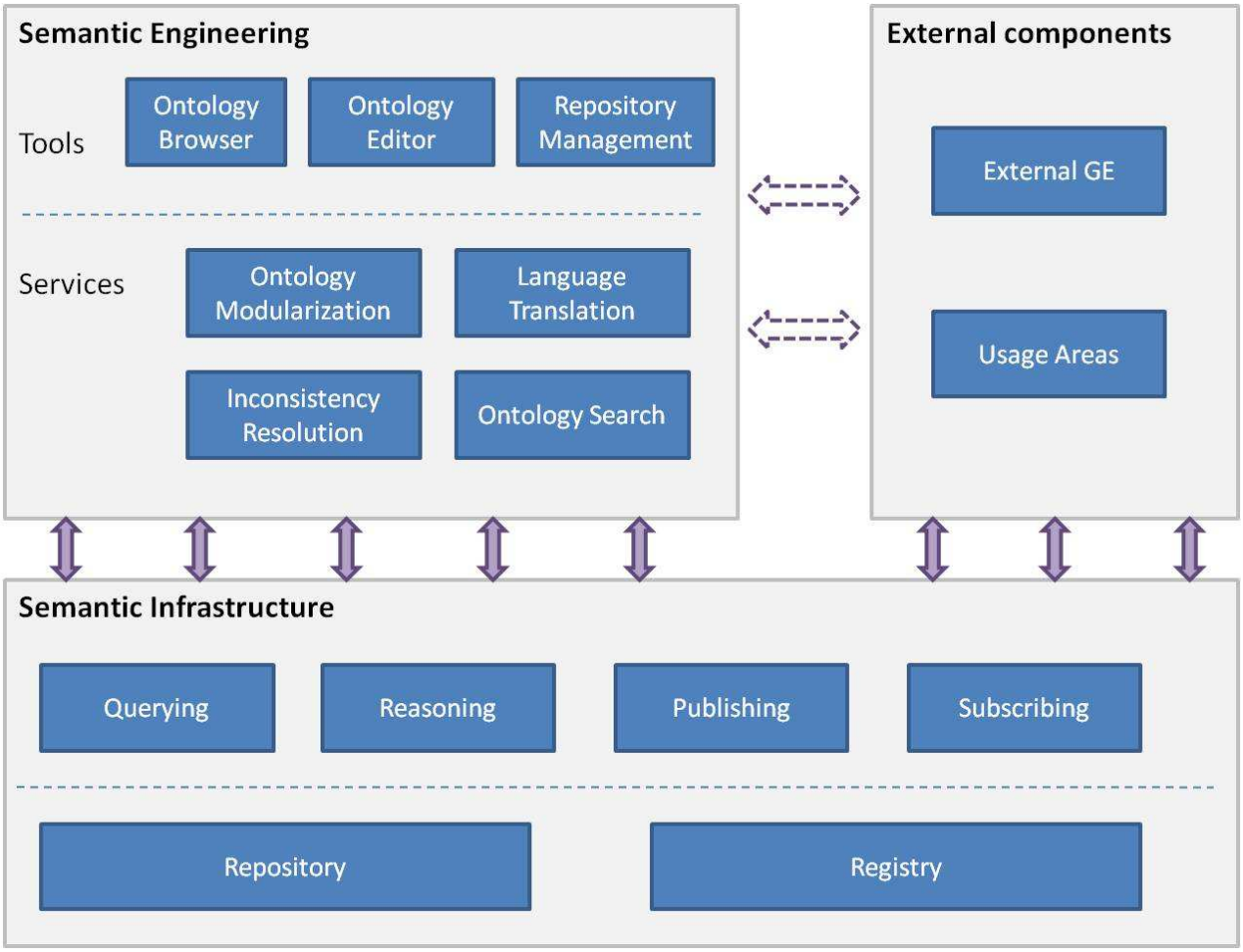


Figure 37 Semantic Web Application Support Enabler architecture

The Semantic Infrastructure is composed of two layers: the storage layer and the utility layer.

The storage layer contains components providing storage for semantic based metadata. Therefore, the repository would storage RDF triples, while the registry would storage ontologies making them available through HTTP protocol. Both repository and registry should meet strong security, scalability and performance requisites in order to support large scale applications.

The utility layer contains components providing business logic that allows applications exploit RDF plus ontologies semantic capabilities. These components include:

- Querying component, that allows the execution of SPARQL queries against a RDF repository.
- Publishing component, that allows the publication of RDF data to a repository.
- Subscribing component, that allows the subscription for specific data.
- Reasoning component, that allows the generation of new knowledge from current knowledge and ontologies.

Due to performance reasons, historically reasoning functionality has been provided by repositories instead of a dedicated component. In the scope of FI-WARE the separation between reasoning and storage component would need further investigations.

The Semantic Engineering is also composed of two layers: the services layer and the tools layer.

The services layer contains services that support processes related with ontology and data engineering such us ontology modularization, ontology translation, ontology search, inconsistency resolutions, etc. As

engineering and methodologies are continually evolving, the services layer will allow the deployment of new services. The interfaces of these services will need further identification.

The tools layer presents a set of tools supporting ontology engineering processes and ontology development methodologies in an integrated way. As methodologies and engineering processes are continually evolving, the tools layer should provide mechanisms to integrate new tools into the current framework. An initial set of tools has been identified:

- The ontology browser, that allows users to navigate and visualize through ontology networks.
- The ontology editor, that allows users to edit ontologies stored in the system.
- The repository management, that allows users to interact with the repository in the Semantic Infrastructure area.

4.2.10.3 *Critical product attributes*

- Provide an infrastructure for semantic web applications that support large scale applications including: metadata storage in RDF, publication of RDF triples, querying by SPARQL and inference.
- Provide a framework for supporting methodologies and engineering processes related with metadata management and ontology development.

4.3 Generic Enablers Implementing Intelligent Services

The Intelligent Services plug-ins interact with the off-line and real-time stream processing enablers, as well as with data that resides in memory and the persistence layer, to provide analytical and algorithmic capabilities in the following main areas: a) Social Network Analysis, b) Mobility Analysis, c) Real-Time Recommendations, d) Behavioural and Web Profiling, and e) Opinion Mining.

These services will be consumed by other FI-Ware components for providing a personalised user interaction, either by adapting the functionality and behaviour based on the user profiles and aggregated knowledge generated (for example the social communities or common itineraries of a user), or by embedding their functionalities into these components (for example displaying recommendations provided by the real-time recommendations service).

4.3.1 Social Network Analysis

4.3.1.1 *GE description*

The Social Network Analysis (SNA) plug-ins analyse the social interactions of users to unveil their social relationships and social communities, and also build a social profile of both individuals and communities. Social Network intelligent services will add the social dimension to the platform, necessary for providing a truly personalised interaction to users.

Data capturing social interactions between users (Facebook posts or comments, re-tweets, SMSs sent and received…) will be processed and analysed to build a network of social connections, including the strength and nature of such connections. The social network derived from interactions will be further analysed to:

- Detect the social communities that appear on the network.
- Build a social profile for the individual user: social connectivity, potential influence on peers, number of communities the user belongs to, etc.

- Build a social profile for communities: number of members, group cohesion, etc.
- Profile the communities across a number of attributes (interests, sociodemographics…) based on the homophily principles that appear in social groups.

Other plug-ins can be provided in this group too, among others, model and predict social diffusion processes, or to provide identity information through the construction of a social fingerprint that can be used for detecting user identity thefts and other security threats.

The SNA plug-ins rely on the following enablers:

- The off-line processing enablers to execute, in a periodic manner, the social network algorithms over the data of social interactions periodically gathered from the various sources available.
- The persistency layer and in-memory storage to store intermediate and final results that will be consumed by other components or plug-ins to provide for example social recommendations.
- The stream processing enablers to make incremental updates to the social network knowledge with low latency.

4.3.1.2 *Critical product attributes*

- The Social Network Analytical capabilities provided by the service integrate social communications of different nature (on-line social networks and mobile communications) to unveil the true social network of users.
- The nature of the communications and their temporal pattern is analyzed to identify truly strong social connections, eliminating noise from the data and finding the social relationships that are really important for users. These relationships are the ones that are relevant for social influence propagation or group behaviours.
- The social communities found on the social network can overlap, i.e., a user can belong to more than one social community e.g. family, friends, colleagues, and they are detected following a scalable and explainable algorithm.
- Community profiles are created across a number of individual attributes like interests or socio-demographic characteristics.
- The influence power calculated for users, which captures the potential influence a user can exert in his social circles, has been proven in a number of business applications.

4.3.2 Mobility Analysis

4.3.2.1 *GE description*

The mobility analysis plug-ins transform geo-located user activity information into a mobility profile of the user. Specifically, geo-located events that contain information about the user generating the event and a timestamp are analysed to extract meaningful patterns of the user's behaviour.

The events processed by the plug-ins must contain longitude and latitude coordinates or have IDs e.g. mobile cell IDs that can be converted into longitude and latitude information through a mobile cell catalogue.

The following information is derived for the user based on the events gathered by the data management platform:

- Points of Interest of the user (home, workplace, usual leisure areas, etc.) and frequent activity there.

- Usual area of activity of the user.
- Frequent itineraries between points of interest.

Other plug-in will receive real-time updates of the user current user-location (longitude-latitude) and will add meaning to this location. The mobility user profile that gets built will be used to provide meaning to the current location of the user, e.g. at home or commuting. This information will be consumed by other services to provide a personalised user interaction based on the current context of the user.

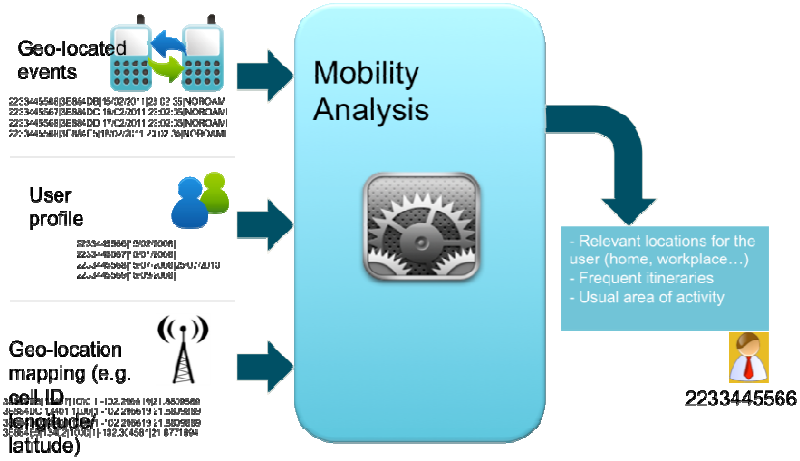


Figure 38 The mobility analysis intelligent services transform geo-located user events into a mobility profile

The mobility analysis plug-ins rely on the following enablers:

- The off-line processing enablers to create the mobility user profiles, as these profiles refer to usual mobility patterns that do not change frequently. Therefore, batch updates will be run, as the expected latency demands do not require stream processing.
- The persistency layer and in-memory storage to store intermediate and final results that will be consumed by other components or plug-ins to, for example, provide communications or a personalised interaction based on the estimated current location of the customer.
- The stream processing enablers to process current user location and provide its meaning to other components.

4.3.2.2 *Critical product attributes*

- Mobility analysis intelligent services transform geo-located user events into a mobility profile that provides a complete view on the usual mobility patterns of the user. This profile is built at a user level and in an automatic fashion, including the detection and labeling of the user points of interest.
- The services go beyond current location data, attaching meaning to real-time locations and transforming it into more actionable and valuable context information.
- The algorithms and models have been used over large volumes of data, and they have been proven to be both efficient and scalable.

4.3.3 Real-time recommendations

4.3.3.1 *GE description*

The real-time recommendation module analyses the behaviour of a user through a service in order to make a recommendation of an item that such a user will most likely be interested in. This can be used in order to increase sales, downloads, or simply to improve the user experience and navigation by showing overall more related content to its interests.

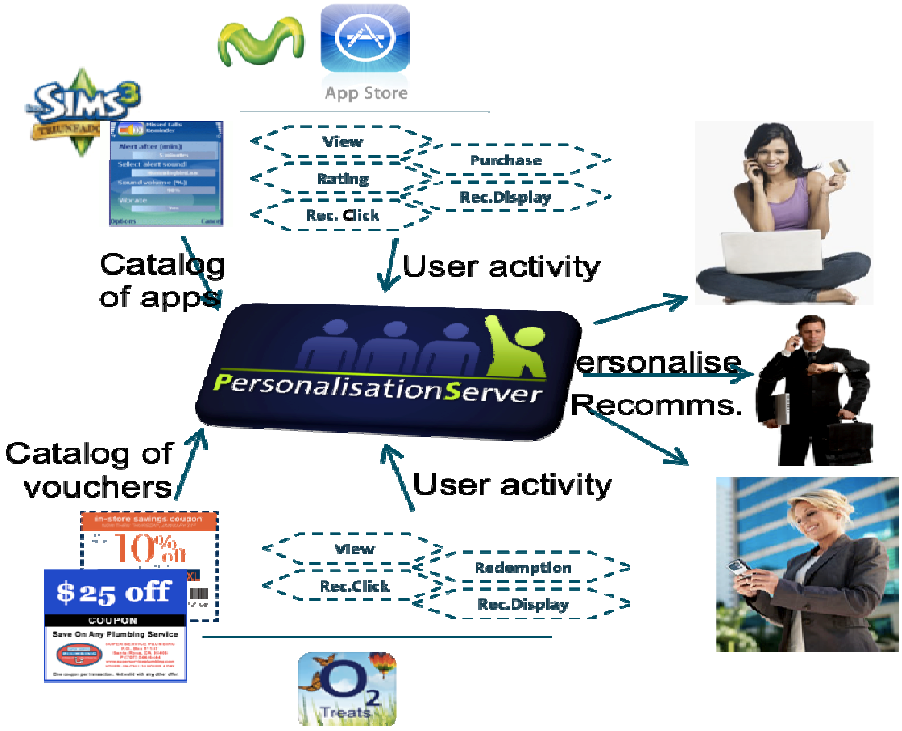


Figure 39 Recommendation system

This module receives two different types of information as input:

- Information from the catalogue of items that will be recommended, such as applications in an application store, ring back tones, music songs or bands, concerts, movies and TV series, etc. Each one of the items in the catalogue needs to have some information describing it such as a title, a unique identifier, possibly a category associated, price and currency if to be sold, a textual description as well as a set of metadata associated to the domain the item belongs to (e.g., device where it runs in the case of applications, actors and director for movies, band and genre for concerts, etc.).
- Activity events generated by users while accessing and interacting with the service and the different items in the catalogue. This information is used in order to compute a profile of the customer as well as a likelihood model to build the recommendations from. Example of types of events include the visualization of an item (showing interest of the user), a rating (explicit feedback), purchases and downloads, etc.

Using this information, this module is able to generate generic (top purchased, top downloaded, most popular, etc.) as well as personalised recommendations to a given user.

4.3.3.2 *Critical product attributes*

- The Recommendations Module provided by the service exploits user activity information in order to create an individual profile of the customer as well as a global profile of the usage of the service
- Such profiles are used in order to generate generic recommendations as well as personalised recommendations for items to be the most relevant for each individual user

4.3.4 Web behaviour analysis for profiling

4.3.4.1 *GE description*

The behavioural and web profiling intelligent service is responsible for the derivation of profiles extracted from the activity of each individual user. Generally speaking, in order to profile a user, it is needed one or more feeds of activity, where different types of feeds might be used, such as

- Transactions of a user within a service (e.g., when the user downloads an element, purchases it, etc.)
- Click-stream within a service (e.g., when a user visualises an item or a classified page of the application)
- Web navigation log (e.g., which web pages the customer is visiting)

Each of these activity feeds might allow for the inference of different information, mostly including

- Categories of interest of a user in a service
- Domain specific information of the activity of a user in a service (e.g., if it is a heavy user, when it uses it, day or night user, devices/channels used to access the service, etc.)
- Main categories visited on the Web, or even main keywords of the web pages visited
- Etc.

4.3.4.2 *Critical product attributes*

- The Behavioural and Web Profiling capabilities provided by the service allows to fully exploit the data generated as part of the interactions of a user with a service.
- These interactions can be analysed and with the right configuration, information is extracted out of it for each individual user, therefore allowing for the inference of an individual user profile extracted purely from the behavior of a user.

4.3.5 Opinion mining

4.3.5.1 *GE description*

Users frequently express their opinions about diverse topics across different channels (blog posts, tweets, Facebook comments, ratings and reviews, calls to customer care…), and this information provides useful insights on both the user and the object of the opinions.

The opinion mining plug-ins provide the analysis of textual sources to derive information about user's opinions, performing the following tasks:

- Language detection.
- Topic detection and classification.
- Sentiment analysis (positive, neutral or negative).

Opinions are provided at three different levels of granularity:

- Opinions of an individual user.
- Aggregated, general opinions
- Structure of the opinions, providing an aggregated view but where different opinion groups e.g. tech-savvy users or elderly people are found based on the source where the opinion was found and the profile of the users expressing the opinion.

For providing an accurate view the bias inherent to the source is taken into account. For example, messages sent to customer care have a negative bias and they express opinions of users of the service or product, while public blog posts can come from non-users. Voice analysis in voice calls – like rate of words/second, pitch, whether certain words are used or not etc – can also be input sources.

The opinion mining plug-ins rely on the following enablers:

- The off-line processing enablers to process the textual sources and perform the required analysis to extract opinions from them.
- The persistency layer and in-memory storage to store intermediate and final results that will be consumed by other components or plug-ins.

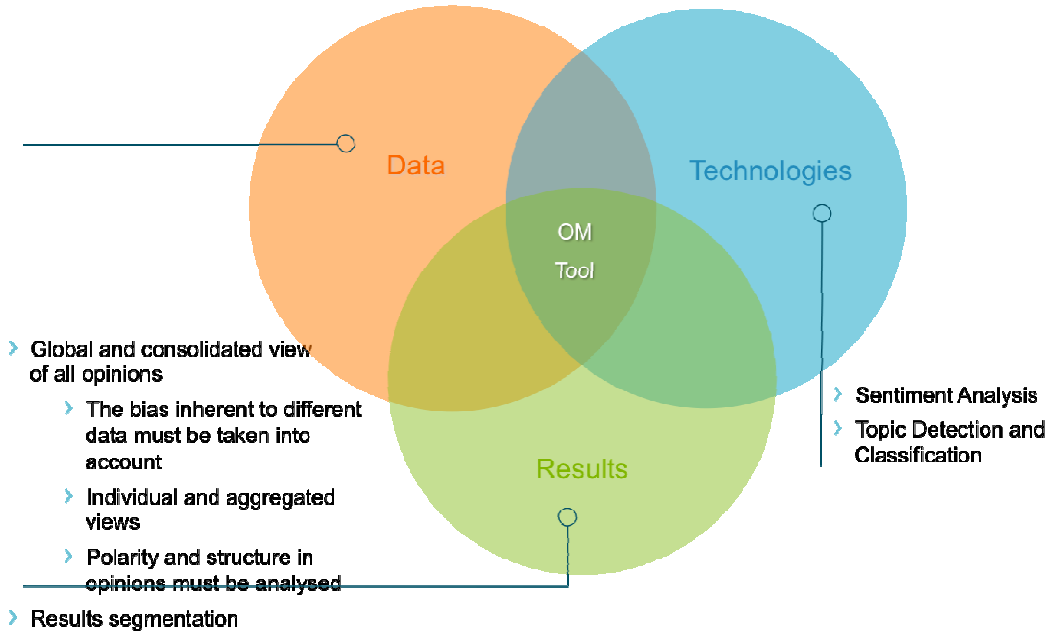


Figure 40 Functionalities of the opinion mining intelligent services

4.3.5.2 *Critical product attributes*

- Textual information from different sources is analysed to extract topics and classify sentiment.
- Both an individual and structured view of opinions from different sources is created, as compared to market solutions that only address the extraction of a single aggregated view of opinions.
- Not all opinion sources and users are treated equally. The services handle the bias inherent to the sources and benefit from the user profile knowledge.

4.4 Question Marks

We list hereafter a number of questions that remain open. Further discussion on these questions will take place in the coming months.

4.4.1 Security aspects

Privacy Management

End users are concerned about privacy of their data. Mechanisms ensuring that applications cannot process data nor subscribe to data without the consent of its owner should be put in place. In addition, end users should be able to revoke access to data they own or establish time limitations with respect to that access. They should be able to mask/partially mask/delete their data. This is a particular function of an obfuscation policy management system. They should also be able to delete data they own as well. Supporting this will require to carefully define the concept of "data ownership" (not only for data provided by end users but generated as a result of processing it), creating mechanisms for establishing and managing different rights based on data ownership.

It should be able to establish privacy control mechanisms enabling enforcement of laws established by the regulatory entities, such as government laws for management of citizens privacy data, or laws establishing rules for management of data privacy within enterprises (e.g., data collected within companies about their employees). This, for example, will warrant that data access rights be set by default at a highest privacy enforcement level, only enabling the customer to reduce control within certain limit (this limit shall be anyway governed for some types of sensitive data such as, e.g., enforced control over minority data).

An mechanism shall be put in place for an automatic control of the privacy issues by, e.g. formal rules (policies) accessed and asserted by data owners and government authorities (regulators) entities and processed by a sort of policy enforcement. It should be able to configure the data (context, events and other including meta-data) that is going to be governed under privacy policies. This mechanism should be able to provide proofs of data privacy policy enforcement

The platform should favour a "privacy by design" approach which in particular makes it possible to deliver/retrieve/store only the necessary data.

Last but not least, provision will be made to manage the re-configurability of security policies in the context of highly dynamic service composition environments. However, the user should be given a consistent view of his attached policies.

Auditing

Access of applications to data should be audited in a secured manner. This may imply that some operations be logged and generated logs be protected. However, audit should take place in a way it doesn't penalize the overall performance.

Transparency of data usage should be ensured: what/who/when data has been collected and how it has been exploited

Trustworthy data

Data may have different level of trustworthiness. Algorithms dealing with analysis of data, and applications in general, should be able to categorize handled data based on trustworthiness level.

A candidate GE is currently under study, which would deal with assessing the quality of information, and in particular the confidence that can be placed in it. Such GE may constitute an essential tool for informational watch, especially in the context of the development of so-called open sources: the increased use of the Web makes it possible for everyone to participate in the information spread and to be a source of information, and thus the quality of information collected on the internet must be assessed. Metadata – typically source reliability but also source certainty – may be processed and analysed by this GE to determine information value. Relationships between information sources, such as affinity and hostility relations, would also be taken into account in order to, e.g., reduce the confirmation effect when sources are known to be friends.

In addition, several aspects should be accountable such as the data storage entity, the data host/provider and the component which exposes some given data.

Data Quality and Control

Data quality concept shall be introduced in FI-WARE in order to allow to the services and applications running on top of FI-WARE capabilities to be selected in and treated accordingly to SLA (Service Level Agreement) and QoS (Quality of Service). However, while the quality of data is optional, its handling within the FI-WARE GE is a mandatory, so that when the QoS/SLA requires certain quality of data and the date are tagged with meta-data about their quality, the FI-WARE shall enable its control and processing respectively.

Identification of Entities and Data

Once data are available within FI-WARE platform they should be univocally identified in order to avoid ambiguity and uncertainty during their handling.

All the entities within FI-WARE shall be unequivocally identified at least within the same application domain in order to process the business logic. In case of coexistence of many service domains within the same application domain, e.g. many different Social Networks providers within the same SN application domain, an arbitrary mechanism, such as a broker, could exist in order to create a strong linking of the same entities even if differently identified, as an alternative to an unique identification among all Social Networks. Then this identification mechanism shall be transparent for the application and services built on top of that domain, e.g. many applications using customers' data from different Social Networks.

Also entities authentication and data usage authorizations can be built and used upon the aforementioned identification technology.

Data Usage Monitoring and Data Access Control

A dedicated mechanism or framework shall be built-in to or enabled by FI-WARE platform that allows the real-time data access control by the entities and data usage management in order to set up the rules for the data management and avoid incorrect data access or alarm data misuse respectively. The framework should support the authentication of the data/context consumer.

Data and Connectivity Availability and Reliability

FI-WARE platform shall integrate or employ mechanisms ensuring data and connectivity availability and reliability at a certain level (e.g. required by SLA or QoS) implemented through "standard" techniques such as redundancy and high-availability. Moreover, if needed by the service or application or by the data nature the overall FI-WARE system shall support resilient mode and fault-tolerance.

The platform should secure flows during data retrieval, in order to preserve data confidentiality, to ensure authenticity of data as well as ensure data integrity. Reliability of the storing of the data.

Non-repudiation of Data Handling and Communications

Non-repudiation property, if required by application, service or by a data nature, shall be supported and provided accordingly by the FI-WARE platform by any available technological means such as e.g. digital certificates and digital signatures.

4.4.2 Other topics still under discussion

Internal GE communication

The method of communication between GEs is under discussion whether it is proprietary and therefore optimized or standard (perhaps using the same interfaces we expect data be accessible to the Context and Data Management and vise versa). For example, what is the method of communication between the Massive Data Gathering GEs and the Processing GEs. What is the method of communication between GEs and the storage for accessing or writing? And how do the processing GEs communicate with the Pub/Sub GE.

When to use Big Data GE and when to use Complex Event Processing GE

Looking closely at the descriptions of the Complex Event Processing GE and the stream processing portion of the Big Data GE they may appear to be addressing the same requirements. Indeed both address the need to continuously process data on the move, data that flows or streams continuously and that sense can be

made out of this data continuously. However they address these requirements differently and for different purposes, thus it is important to describe the commonalities, differences and probably more important to give insight which GE to use, when and for what purpose.

The most important distinction between the two approaches is the programming paradigm. CEP takes more of a declarative (rule-based) approach whereby developers of event processing applications have all the necessary constructs to declare the necessary processing without programming and can therefore be targeted to more business oriented users and suited for higher rates of change of the processing logic. Stream processing is more algorithmic, requiring programming skill and is therefore targeted to IT programmers, where the processing logic is programmed, assembled, compiled and then deployed.

Another distinction is in performance aspects and how the two approaches are designed to address them. Generally, the stream processing approach is designed to cope with very high rates of receiving data usually more than the CEP approach. The latency in generating results is usually lower in stream processing than CEP. This comparison, however, is not based on executing the same processing logic in the two approaches, rather it is based on the nature of the scenarios the approaches are applied to.

Additional insights [Chandy11] to be aware of:

- Complex pattern matching is common in CEP but it is generally not central to stream processing.
- Stream processing queries tend to be compositions of database-style operators (e.g., joins) and user-defined operators. These queries mostly represent functionality of aggregation and transformation.
- Stream processing tends to place a higher emphasis on high data volumes with relatively fewer queries.
- CEP tends to consider the effect of sharing events across many queries or many patterns as a central problem.
- Processing unstructured data is typically not considered a scenario for CEP.
- Streaming systems, having originated mainly from the database community, tend to have a schema that is compatible with relational database schemata, whereas CEP systems support a larger variety of schema types.

Data gathering

In Figure 13, a Massive Data Gathering Enabler is depicted that collects data from different sources. At the moment, this Generic Enabler is not covered (fully) by the GEs listed under section 4.1. However, work to fill this gap is taking place. There are mainly two GEs that already provide part of the functionality. These GEs can be seen as Transformation Enablers (e.g., realised as plug-ins) to the Massive Data Gathering Enabler.

The GE on pre-processing of meta-data (section 4.2.6) transforms and filters incoming meta-data in order to prepare the data for subsequent processing. E.g., a transformation of inbound meta-data to (Java) classes can be performed to supply the correct input format for a stream processing engine requiring a dedicated meta-data format.

The GE on pre-processing of unstructed data (section 4.2.5) works on data without explicit semantic or formatting in contrast to the above enabler. The semantic and ontology of these kinds of input data are derived during the operations inside the GE, e.g., data cleaning, ontology evolution, information extraction, and sentiment analysis.

The two Gerneric Enablers could even interwork inside a Massive Data Gathering Enabler, i.e., the GE on pre-processing of unstructed data serving as input to the GE on pre-processing of meta-data. A final version of the Massive Data Gathering Enabler is envisioned as a generic sink/collector for various kinds of data sources in order to provide input for subsequent processing like Complex Event Processing (section 4.2.2) and Big Data Analysis (section 4.2.3).

Query Broker

The Query Broker GE targets the ambitious goal of being capable to handle queries formulated in any of a defined set of query languages/APIs (e.g., XQuery, SQL or SPARQL). This requires all incoming queries to be converted into an internal abstract format that in turn be translated into the respective specific query languages/APIs supported by the actual data repositories. This also requires to deal with different retrieval systems, not always following the same data retrieval paradigms. A careful analysis should be made on whether this GE can be recommended for general usage or just as a mechanism enabling federation of multiple and heterogeneous data sources, involving multiple data storage formats and retrieval paradigms, when usage scenarios require such federation. The former case would require that execution of queries expressed in a given language do not suffer any performance penalty when issued to a data repository natively supporting that query language.

The Publish/Subscribe Broker GE export operations for querying data/context elements being generated by Event Producers. It would be highly desirable that specification of these operations are aligned to those exported by the Query Broker GE. This would allow, among other things, treating Publish/Subscribe GEs as data sources which may be federated with other data sources. However, this requires further analysis.

Last but not least, the question about how non-relational or "NoSQL" databases, which are becoming an increasingly important part of the database landscape, can be integrated, is also one of the open points to be addressed during the FI-WARE project.

Localization services

Some topics has been identified for further discussion regarding capabilities supported by the Localization GE beyond those related to support of SUPL and MLP: RRLP protocol, timing advance info from base stations, base station visibility and RX level gained from SIM/ME interface etc. It has also been pointed out that this GE should not be a centralized GE, but a distributed one, that has components running inside smart/feature phones and/or SIM cards. Yet additionally, this should have optionally a "proactive" feature, determining devices to "make noise" if an application wants more accurate location information about them.

The current Localization GE establishes MLP as the basic interface that applications may use to retrieve the current position of a mobile device or get event reports linked to variation of these positions (when the device enters, leaves, remains inside, or remains outside a given area). However, discussion is taken place to design how an instance of the Publish/Subscribe Broker GE may be connected to the Localization GE so this information may be delivered in the form of context events. This has the advantage of being able to merge handling of location-related events with handling of any other kind of context and even data events relevant to a given application. In addition, it would allow to setup parameters for the reporting of location events linked to a given set of mobile devices that would be common to several applications while still allowing each application to handle its subscription to location events, changing it dynamically without affecting others. Last but not least, it may also make it easier to forward location events to a Complex Event Processing GE or BigData Analysis GE.

Other topics still under discussion regarding the Localization GE have to do with enriching its functions as to include location propagation and learning features. Location propagation features are used when location of a desired mobile device is unknown and cannot be retrieved, while that device is in a detectable proximity with another entity (e.g., mobile device of another user, thing that can be detected as near) which location is known and could be retrieved. In this case, the location of the mobile device is obtained through the propagation of the location of the entity (or some location calculated as adjustment of the location of the entity). Learning features require registering locations of mobile devices as they are resolved and location of entities based on some application interactions (e.g., check-in of users in restaurants). Data/context that is learned can then be further exploited to enhance location of mobile devices and entities by the SLP, including the enhancement of propagation techniques.

4.5 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)

- **Data** refers to information that is produced, generated, collected or observed that may be relevant for processing, carrying out further analysis and knowledge extraction. Data in FI-WARE has associated a **data type** and a **value**. FI-WARE will support a set of built-in **basic data types** similar to those existing in most programming languages. Values linked to basic data types supported in FI-WARE are referred as **basic data values**. As an example, basic data values like '2', '7' or '365' belong to the integer basic data type.
- A **data element** refers to data whose value is defined as consisting of a sequence of one or more <name, type, value> triplets referred as **data element attributes**, where the type and value of each attribute is either mapped to a basic data type and a basic data value or mapped to the data type and value of another data element.
- **Context** in FI-WARE is represented through **context elements**. A context element extends the concept of **data element** by associating an EntityId and EntityType to it, uniquely identifying the entity (which in turn may map to a group of entities) in the FI-WARE system to which the context element information refers. In addition, there may be some attributes as well as meta-data associated to attributes that we may define as mandatory for context elements as compared to data elements.

  Context elements are typically created containing the value of attributes characterizing a given entity at a given moment. As an example, a context element may contain values of some of the attributes "last measured temperature", "square meters" and "wall color" associated to a room in a building.

  Note that there might be many different context elements referring to the same entity in a system, each containing the value of a different set of attributes. This allows that different applications handle different context elements for the same entity, each containing only those attributes of that entity relevant to the corresponding application. It will also allow representing updates on set of attributes linked to a given entity: each of these updates can actually take the form of a context element and contain only the value of those attributes that have changed.
- An **event** is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. Events typically lead to creation of some data or context element describing or representing the events, thus allowing them to processed. As an example, a sensor device may be measuring the temperature and pressure of a given boiler, sending a context element every five minutes associated to that entity (the boiler) that includes the value of these to attributes (temperature and pressure). The creation and sending of the context element is an event, i.e., what has occurred. Since the data/context elements that are generated linked to an event are the way events get visible in a computing system, it is common to refer to these data/context elements simply as "events".
- A **data event** refers to an event leading to creation of a data element.
- A **context event** refers to an event leading to creation of a context element.
- An **event object** is used to mean a programming entity that represents an event in a computing system [EPIA] like event-aware GEs. Event objects allow to perform operations on event, also known as **event processing**. Event objects are defined as a data element (or a context element) representing an event to which a number of standard event object properties (similar to a header) are associated internally. These standard event object properties support certain event processing functions.

4.6 References

[Craswell 99]	Nick Craswell, David Hawking, and Paul B. Thistlewaite, “Merging results from isolated search engines,” in Proceedings of the Australasian Database Conference, 1999, pp. 189–200.
[Döller 08a]	Mario Döller, Ruben Tous, Matthias Gruhne, Kyoungro Yoon, Masanori Sano, and Ian S Burnett, “The MPEG Query Format: On the way to unify the access to Multimedia Retrieval Systems,” IEEE Multimedia, vol. 15, no. 4, pp. 82–95, 2008.
[Döller 08b]	Mario Döller, Kerstin Bauer, Harald Kosch, and Matthias Gruhne, “Standardized Multimedia Retrieval based on Web Service technologies and the MPEG Query Format,” Journal of Digital Information, vol. 6, no. 4, pp. 315–331, 2008.
[Döller 10]	Mario Döller, Florian Stegmaier, Harald Kosch, Ruben Tous, and Jaime Delgado, “Standardized Interoperable Image Retrieval,” in ACM Symposium on Applied Computing (SAC), Track on Advances in Spatial and Image-based Information Systems (ASIIS), Sierre, Switzerland, 2010, pp. 881–887.
[EPTS]	Event Processing Technical Society: http://www.ep-ts.com/
[EPTS-RA 10]	Adrian Paschke and Paul Vincent. “Event Processing Architectures”, Tutorial, DEBS 2010. http://www.slideshare.net/isvana/debs2010-tutorial-on-epts-reference-architecture-v11c
[ISO 06]	ISO/IEC 23001-1:2006, Information technology – MPEG system technologies – Part 1: Binary MPEG Format for XML, Apr. 2006.
[ISO 08]	ISO/IEC 14496-12:2005 Information technology – Coding of audio-visual objects – Part 12: ISO base media file format, third edition, Oct. 2005.
[OMA-TS-NGSI-Context]	OMA-TS-NGSI_Context_Management-V1_0-20100803-C. Candidate Version 1.0. 03, August 2010. Open Mobile Alliance. http://www.openmobilealliance.org/Technical/release_program/docs/CopyrightClick.aspx?pck=NGSI&file=V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf
[RFC3550]	H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A transport protocol for real-time applications,” RFC 3550, Jul. 2003.
[RFC3984]	S. Wenger, M. M. Hannuksela, M. Westerlund, and D. Singer, “RTP payload format for H.264 video,” RFC 3984, Feb. 2005.
[Smith 08]	John R. Smith, “The Search for Interoperability,” IEEE Multimedia, vol.15, no. 3, pp. 84–87, 2008.
[W3C 11]	W3C, “Efficient XML Interchange (EXI) Format 1.0”, Mar. 2011

5 Applications/Services Ecosystem & Delivery

5.1 Overview

The Application and Services Ecosystem and Delivery Framework in FI-WARE comprises a set of generic enablers (i.e. reusable and commonly shared functional building blocks serving a multiplicity of usage areas across various sectors) for creation, composition, delivery, monetisation, and usage of applications and services on the Future Internet. It supports the necessary lifecycle management of services and applications from a technical and business perspective, the latter including business aspects such as the management of the terms and conditions associated to the offering, accounting, billing and SLAs, thus enabling the definition of a wide range of new business models in an agile and flexible way along with their association with the different application and services available in the ecosystem. The capacity to monetize applications and services based on those business models, adapting the offering to users and their context and dealing with the fact that they may have been built through the composition of components developed and provided by different parties is a key objective for the **business framework** infrastructure.

FI-WARE Apps/Services delivery framework considers the ability to access and handle services linked to processes, 'things' and contents uniformly, enabling them to be mashed up in a natural way. The framework brings the necessary **composition and mash-up** tools that will empower users, from developers to domain experts to citizens without programming skills, to create and share in a crowd-sourcing environment new added value applications and services adapted to their real needs, based on those offered from the available business frameworks. A set of **multi-channel/multi-device adaptation** enablers are also contributed to allow publication and delivery of the applications through different and configurable channels, including those social web networks which may be more popular at the moment, and their access from any sort of (mobile) device.

The high-level architecture illustrated in Figure 41 is structured according to the key business roles and their relationships within the overall services delivery framework and existing IT landscapes. The technical architectures implementing these business roles and relationships are more complex and will be discussed and illustrated in more detail in the subsequent sections. The applications and services delivery framework comprises the internal key business roles: Aggregator, Broker, Gateway, and Channel Maker. Furthermore, there are the external key roles: Provider, Hoster, Premise, and Consumer.

The ***Provider*** role supports partners that hold governance and operational responsibility for services and apps through their business operations and processes. The Provider role allows services and applications to be exposed into a business network and services ecosystem so that they can be accessed without compromise to the given delivery mechanisms. For example, a provider should be able to publish a service to a third-party, while still requiring that it run through its current hosting environment and be securely interacted with. Some providers may have services to be re-hosted onto a third-party cloud or to be downloaded and running in the users' environment. Given that the service will be accessed by different parties, the provider needs to ensure that the service is comprehensively described to ensure it is properly used, especially for the benefit of consumers. Thus, for example the description of a service has to include: data about service ownership, functional descriptions, dependencies on other services, pricing, and consumer rights, penalties and obligations in terms of service performance, exceptions, information disclosure and other legal aspects. For services related in wider business contexts, domain-specific semantic descriptions, vertical industry standards and other documents containing service-related information (like policy, legislation and best practices documents) are also useful to enhance service discovery and consumer comprehension of services. Therefore, they need to be linked to service descriptions and used during service discovery and access through unstructured search techniques.

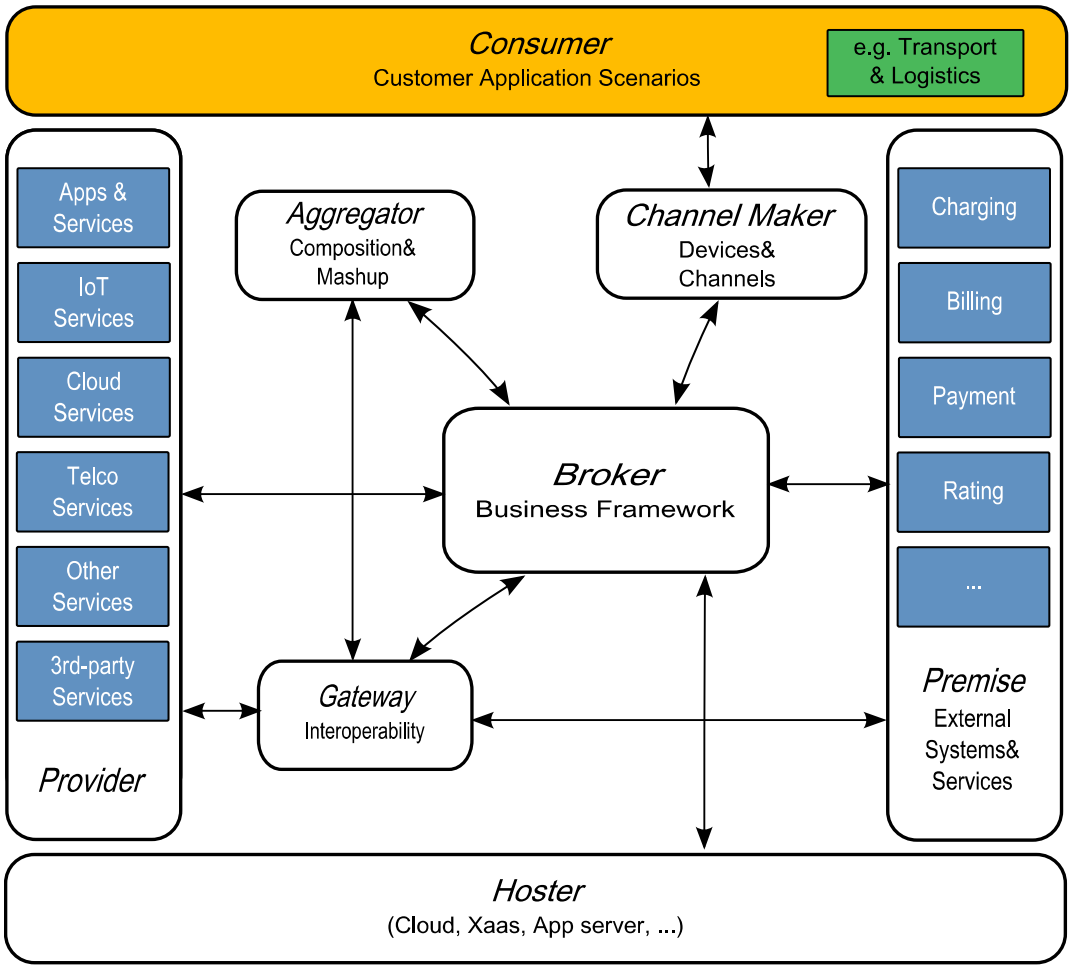


Figure 41 High-level architecture

The ***Broker*** role supports exposing services from diverse providers into new markets, matching consumer requests with capabilities of services advertised through it. The Broker will provide the Business Framework consisting of the "front-desk" delivery of services to consumers and a flexible set of "back-office" support for the secure and trusted execution of the service. The Broker is central to the business network and is used to expose services from providers, to be onboarded and delivered through the Broker's service delivery functionality (e.g. run-time service access environment, billing/invoicing, metering). The Provider interacts with the Broker to publish and register service details so that they can be accessed through the Broker. Services consumed through end users or applications, or services offered in a business network through intermediaries can be published, alike, in the Broker. Third-parties can extend and repurpose services through the Broker, by using the functionality of the Broker to discover, contract and make use of services through design-time tooling (see subsequent sections for further details). Ultimately service delivery is managed through the Broker when services are accessed at run-time by end users, applications, and business processes. In short, the Broker provides a *monetization* infrastructure including directory and delivery resources for service access.

Although services are accessed through a broker, the execution of their core parts resides elsewhere in a hosted environment. Certain types of services and apps could be re-hosted from their on-premise environments to cloud-based, on-demand environments to attract new users at much lower cost, without the overheads of requiring access to large application backends.

The ***Hoster*** role allows representing the different cloud hosting providers involved as part of the provisioning chain of an application in a business network. An application/service can be deployed onto a specific cloud using the Hoster's self-service interface. By providing a generic interface for Hosters to

report Usage Accounting Records, FI-WARE opens up the possibility to access several cloud providers and implement interesting revenue share models between Providers and Hosters. By defining a standard self-service interface, in turn, FI-WARE opens up the possibility to migrate among alternative FI-WARE Cloud Instance Providers, which would play the Hoster role, avoiding "lock-in". Work in the Apps and services delivery framework chapter will closely interact with work in the Cloud Hosting chapter in order to address the Hoster interfaces topics (Usage Accounting Records reporting and Self-service interfaces). Cloud services are highly commoditized with slim margin and are subject to business volatility. Cloud services exposed to a business network should be advertised through the Broker. When a service needs to be hosted, the Broker can help to match its hosting needs (e.g. platform services, operating systems) with cloud services (advertised through the Broker). The Broker performs the matching and lists candidate cloud services for a user (a provider requiring hosting as a part of exposing a service offer in a business network or a consumer negotiating hosting options/costs when a service is ordered). Once hosted, the cloud service may be monitored for performance and reliability. FI-WARE Cloud Instance Providers playing the role of Hosters offer business networks partners the possibility to shift cloud providers efficiently, avoiding being "lock-in" a concrete Cloud Hosting Provider.

The ***Aggregator*** role supports domain specialists and third-parties in aggregating services and apps for new and unforeseen opportunities and needs. Application services may be integrated into corporate solutions, creating greater value for its users not otherwise available in their applications, or they could be aggregated as value-added, reusable services made available for wider use by business network users. In either case, the Aggregator provides the dedicated tooling for aggregating services at different levels - UI, service operation, business process or business object levels.

Aggregators are similar to providers however, there is an important difference. Aggregators do not operate all services that they aggregate. Rather, they obtain custodial rights from providers to repurpose services, subject to usage context, cost, trust and other factors. Despite new aggregated services being created, constituent services execute within their existing environments and arrangements: they continue being accessed through the Broker based on a delivery model (as discussed above). The delivery framework offers service-level agreement support through its different roles so that providers and aggregators can understand the constraints and risks when provisioning application/services in different situations, including aggregation of services.

The Aggregator provides design-time and run-time tooling functionality. It interacts with the Broker for discovery of services to be aggregated and for publishing aggregated services for use by the business network partners.

The ***Gateway*** role supports Providers and Aggregators in selecting a choice of solutions that may provide interoperability, as a service, for their applications. This can include design-time business message mapping as well as run-time message store-forward and message translation from required to proprietary interfaces of applications. This is beneficial when services need to be exposed on a business network so that they can be accessed by different parties. When a provider exposes a service onto a business network, different service versions can be created that have interfaces allowing interactions with different message standards of the partners that are allowed to interact with the service.

The gateway services are advertised through the Broker, allowing providers and aggregators to search for candidate gateway services for interface adaptation to particular message standards. Key differentiators are the different pricing models, different communities engaging in their hubs, and different quality of services. The gateway services may address design time mappings as well as run-time adaptation.

The ***Channel Maker*** role provides support for creating outlets through which services are consumed. Channels, in a broad sense, are resources, such as Web sites/portals, social networks, mobile channels and work centres, through which application/services are accessed. The mode of access is governed by technical channels like the Web, mobile, and voice response.

The notion of channelling has obvious resonance with service brokerage. Virtually all the prominent examples of Web-based brokers like iTunes, eBay and Amazon expose services directly for consumption and can also be seen as channels. The service and apps delivery framework's separate designations of the service channel and the service broker addresses a further level of flexibility for market penetration of

services: service channels are purely points of service consumption while brokers are points of accessing services situated between channels and hosting environments where the core parts of service reside. This separation allows different service UIs and channels to be created, outside the capacity of those provided by brokers. In fact, different channels can be created for services registered through the same broker. The delivery framework, in other words, allows for consolidation of service discovery and access through the Broker and independent consumption points through different channels enabled through the Channel Maker. This is especially useful for mainstream verticals like the public sector dedicating whole-of-government resources for service discovery and access, but requiring a variety of channels for its different and emerging audiences.

The creation of a channel involves selection of services consumed through a channel, constraining which service operations are to be used, what business constraints apply (e.g. availability constraints), and how the output of an operation should be displayed (forms template) in the channel. The Channel Maker interacts with the Broker for discovery of services during the process of creating or updating channel specifications as well as for storing channel specifications and channelled service constraints back in the Broker.

The ***Consumer*** role completes the service supply chain, effectively fostered by the delivery framework. Through the Consumer, parties can manage the "last mile" integration where application and services are consumed through different environments. This involves the allocation of resources consuming services to consumer environments in which they operate and the interfacing of consumer environments so that the services required can be accessed at run-time in a secure and reliable way.

As discussed above, the resources that consume services are either explicit channels or applications that have services integrated ("hard-wired" into code) or accessible (through a discover/access interface). The Channel Maker and Aggregator are used for supporting the processes of integration of channels and applications, respectively. Since the allocation of applications in organizations is a specialized consideration controlled through administration systems, the Consumer is used mostly for allocating channels in consumer environments (inside organizations or wider availability on the Web).

The Consumer supports consumer environments so that the services they require are integrated with the Broker through inbound and outbound run-time interactions. Recall, the Broker allows services to be accessed through different delivery models. Interfaces are required on remote consumer environments so that, on the one hand, applications running in the environments have well-defined operations that allow interactions with the Broker. Channels can be discovered through the Broker and allocated through the Consumer for usage through designated users/groups in the business network.

In the following sections we describe the architectures realizing the introduced roles in more detail and identify generic enablers. The sections are organized in the following way:

- The business framework infrastructure realizes the Broker functionality
- Composition and Mashup infrastructure covers the Aggregator and Gateway functionality
- The Channel Maker is addressed by Multi-Service Multi-device adaptation

5.2 Generic Enablers of the Business Framework

Figure 42 illustrates the high-level architecture of the business framework infrastructure realizing the Broker. The architecture identifies the following core components of the business framework: Marketplace, Shop, USDL Repository and Registry, Business Elements and Models Provisioning System (BE&BM Provisioning), Revenue Sharing Engine, and SLA Management. Furthermore, it shows the core relationships to external parties and systems necessary to make the business framework infrastructure operational.

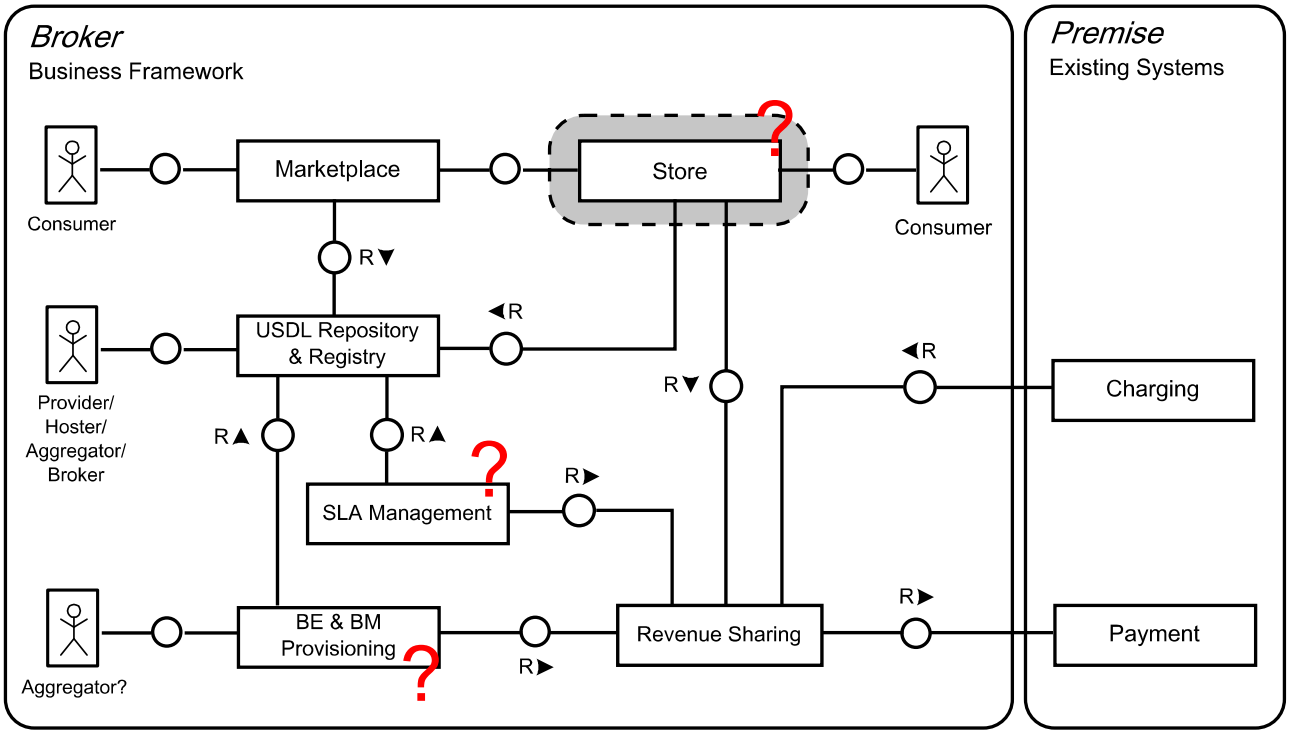


Figure 42 High-level architecture of the Business Framework[11]

The set of Generic Enablers (GE) identified are:

[11] The red question marks indicate issues discussed in the question marks section (5.6)

- **USDL Repository:** The service repository is a place to store service descriptions or parts of it. A location for storage (centrally or distributed and replicated), for reference and/or safety. The use of a repository is required in order to prepare service descriptions to appear at a store, marketplace and other components of the business framework.
- **USDL Registry:** The registry is a universal directory of information used for the maintenance, administration, deployment and retrieval of services in the service delivery framework environments. Existing (running) service endpoints as well as information to create an actual service instance and endpoint are registered. The USDL Registry links USDL descriptions in the USDL repository with technical information about instances available in the platform. Similar to UDDI Registry for web services it is a place to find information for technical integration. Only if the service endpoint is registered it actually can be used for service composition and coordination by the FI-Ware platform.
- **Marketplace and Store:** We differentiate the marketplace from a store in our architecture. While a store is owned by a store owner who has full control over the specific (limited) service/app portfolio, and offerings a marketplace is a platform for many stores to place their offerings to a broader audience and consumers to search and compare services and find the store, where to buy. The final business transaction (buying) is done at the store and the whole back office process is handled by the store. There are existing Internet sales platforms that actually have marketplace and store functionality combined. However, conceptually the distinction is useful in order to simplify the architecture and have a better separation of concerns. Due to a large variety of already existing stores and offered functionalities, the store is considered as external component to be integrated with the business framework infrastructure. The main focus will be on developing a secure marketplace as a generic enabler and providing interfaces to the store.
- **Business Elements and Models Provisioning System:** The aim of the BE&BM Provisioning is the monetization of services, applications, and their compositions/aggregations. It is necessary to have a flexible way to define the manner in which services and applications can be sold and delivered to the final customers; it can be summarized as the business model definition. While the published USDL description represents the public view of the business model offered to the customer, the business model defines the way in which customers pay by application and services and the way in which the incomes are to be splitted among the involved parties. Once, the business model is defined, it is necessary to provision these details in the rating/charging/billing systems.
- **Revenue Settlement and Sharing System:** In the Future Internet there is a need to manage in a common way how to distribute the revenues produced by a user's charges for the application and services consumed. When a consumer buys/contracts an application or service, he pays for its usage. This charge can be distributed and split among different actors involved (for instance store or marketplace owner earns money and mash-ups have to split the money). There will be a common pattern for service delivery in service-oriented environments. Independent of service type, composite services based on the aggregation of multiple atomic (from the viewpoint of composition) services are expected to play an important role in applications and services ecosystems. Beyond the complexities of the management of composite services (design, provisioning, etc.), there is a complex issue to solve when dealing with the business aspects. Both the composite and the atomic services must be accounted, rated and charged according to their business model, and each of the service providers must receive their corresponding payment. The Revenue Settlement and Sharing System serves to the purpose to split the charged amounts and revenues among the different services providers.
- **SLA Management:** The management of Service Level Agreements (SLAs) will be an essential aspect of service delivery in the future internet. In a competitive service market place, potential customers will not be looking for "a" service, but for "the best" service at the "best price". That is, the quality of services (QoS) – such as their performance, economic and security characteristics - are just as important, in the market place, as their functional properties. Providers who can offer hard QoS guarantees will have the competitive edge over those who promote services as mere 'functional units'. SLAs provide these hard guarantees: they are legally binding contracts which specify not just that the provider will deliver some service, but that this service will also, say, be delivered on time, at a given price, and with money back if the pledge is broken. The cost of this increased quality assurance,

however, is increased complexity. A comprehensive and systematic approach to SLA management is required to ensure this complexity is handled effectively, in a cohesive fashion, throughout the SLA life-cycle.

5.2.1 USDL Service Descriptions

5.2.1.1 *Target usage*

The Unified Service Description Language (USDL) is a platform-neutral language for describing services. It was consolidated from SAP Research projects concerning services-related research as an enabler for wide leverage of services on the Internet. With the rise of commoditized, on-demand services, the stage is set for the acceleration of and access to services on an Internet scale. It is provided by major investments through public co-funded projects, under the Internet of Services theme, where services from various domains including cloud computing, service marketplaces and business networks, have been investigated for access, repurposing and trading in large settings (e.g., FAST,[12] RESERVOIR,[13] MASTER,[14] ServFace,[15] SHAPE,[16] SLA@SOI,[17] SOA4ALL[18]), and the Australian Smart Services CRC.[19])

The kinds of services targeted for coverage through USDL include: purely human/professional (e.g. project management and consultancy), transactional (e.g. purchase order requisition), informational (e.g. spatial and demography look-ups), software component (e.g. software widgets for download), digital media (e.g. video & audio clips), platform (e.g. middleware services such as message store-forward), security and infrastructure (e.g. CPU and storage services).

User roles

- Service providers are describing all aspects of the service from their business point of view.
- Brokers search and use the USDL information for filtering, aggregation and bundling of services.
- Consumers can search and read information from USDL descriptions indirectly via the marketplace user interface.
- Shop owners to specify their offerings on the marketplace.
- Marketplace owner to do a comparison of service offerings.

[12] http://fast-fp7project.morfeo-project.org/
[13] http://www.reservoir-fp7.eu/
[14] http://www.master-fp7.eu/
[15] http://141.76.40.158/Servface/
[16] http://www.shape-project.eu/
[17] http://sla-at-soi.eu/
[18] http://www.soa4all.eu/
[19] http://www.smartservicescrc.com.au/

A generic service description language for domains as diverse and complex as banking/financials, healthcare, manufacturing and supply chains, is difficult to use and therefore not sufficient. First of all, not all aspects of USDL apply to all domains. Rather, USDL needs to be configured for the particular needs of applications where some concepts are removed or adapted while new and unforeseen ones are introduced. A particular consideration of this is allowing specialized, domain-specific classifications such as those available through vertical industry standards to be leveraged through USDL. In addition to this, the way in which USDL is applied for deployment considerations, e.g., the way lifecycle versioning applies, needs to be managed without compromising the fundamental concepts of USDL. In other words, USDL needs to be applied through a framework which allows separation of concerns for how it is applied and tailored to concrete applications. This need has led to the USDL framework where the concepts of the USDL meta-model as a core are specialized through the USDL Application meta-model. A non-normative specialization of the USDL meta-model with the USDL framework is provided to illustrate how a service directory of a specific Service Delivery Framework (proposed by SAP Research) can be conceptualized through USDL. In this way, an insight is available for an application of USDL.

To make FI applications and services more widely available for such composition and consumption, a uniform standardized way of describing and referencing them is required. A variety of service description efforts has been proposed in the past. However, many of these approaches (e.g. UDDI, WSMO, or OWL-S) only prescribe tiny schemata and leave the modelling of service description concepts such as a generic schema for defining a price model or licenses to the service developer.

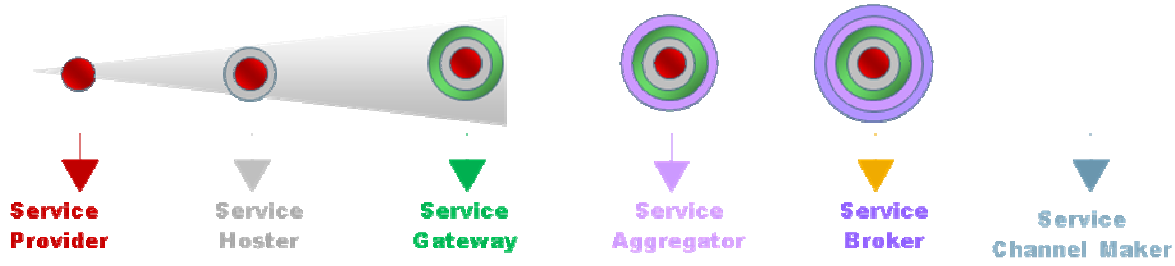


Figure 43 Roles of participants in the Internet of Services Ecosystem

5.2.1.2 *GE description*

The FI-Ware approach will allow comprehensive service descriptions by employing the Unified Service Description Language (USDL) in its registry and repository. USDL builds on standards for the technical description of services, such as WSDL, but adds business and operational information on top. With its ability to describe both human and IT-supported services that not only implement business processes, but also tie in assets linked to contents and the Internet of Things, USDL is set apart from many of the related approaches mentioned above.

USDL is modularized to describe various aspects of services:

- Foundational Module: This module provides a common set of concepts and properties, such as time, location, organization, etc. that are used in all modules.
- Service Module: Describes the general information about the service type, nature, titles, taxonomy and descriptions.
- Participant Module: This module describes the participating organizations, contact persons and their role within the service fulfilment.
- Functional Module: This module contains information about the specific capabilities of a service, input/output parameters and constraints.
- Interaction Module: A module that describes the points of interaction and the responsible participants or participant roles in course of the service fulfilment.

- Technical Module: Describes how functions (capabilities) of a service are mapped to technical realizations of the service (e.g. WSDL operations, parameters, faults, etc.)
- Pricing Module: Contains information about price plans, price components, fences, etc. for a service.
- Service Level Module: The module which specifies service level agreements, such as time schedules, locations, and other constraints.

Legal Module: Contains information about the terms and conditions, IPR, licenses, and rights of use for the participating parties.

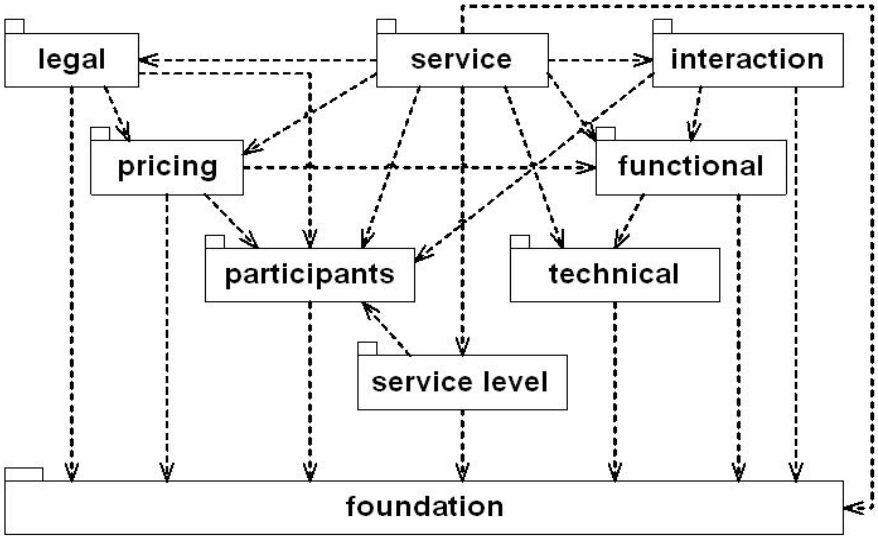


Figure 44 USDL modules and relationships

Necessary extensions to USDL for business, operational and technical perspective of supported front-end mashups and backend composite services will be identified and defined in the context of FI-WARE project. Furthermore, USDL extensions for artefacts from the chapters listed below will be considered:

- Internet of Things Service Enablement

  Applications and services may have specific requirements of handling access to and management of sensor data and things within the "real" world. In many application scenarios, such as logistics, retail, or manufacturing, these capabilities will have an impaction even on business, as well as operational aspects. It will be necessary to investigate the extension of USDL according to aspects of the service lifecycle, which allows describing the relevant properties and capabilities of the Internet of Things in respect to accessing data as well as managing resources.

- Security, Privacy, Trust

  The future internet platform will heavily depend on security, privacy and trust in any business relevant activity. Especially crowd-sourcing, flexible composition and mashups require adequate mechanisms to maintain security related characteristics. At the business framework level it is necessary to make security-specific information transparent. USDL needs to be extended by modules that allow describing the security related qualities and requirements of services and applications.

- Cloud Hosting (XaaS monetization)

  Many services/apps will rely on specific hosting and cloud services, which will have technical as well as business relevant implications. The service description needs to contain the relevant information on

hosting in order to cover these implications. Therefore XaaS description modules for describing various hosting modules need to be provided.

- Data/Context Management Services

  For security monitoring and analytics of activities in the platform it will be necessary to emit and collect events and data for real-time or post-mortem analysis. For the various stakeholders it need to made transparent, when, how, and which data will be collected for which purpose and to whom. So the service description needs to be enriched by data and context information accordingly.

USDL will provide the means for integration of all these areas, FI-Ware generic enablers and the respective platform products. It allows tracing information and processes across the whole service lifecycle.

5.2.1.3 *Critical product attributes*

- rely on Web standards such as HTML, XML, RDF, …
- openness integrate into different contexts
- extensibility for addressing unforeseen or domain specific aspects
- flexibility to adapt to different domains via module variants
- harmonized with other information descriptions on the Web (existing schemas)
- ability to link-up with other information on the Web
- simplicity, easy to understand
- expressive power to specify at an appropriate level
- graceful degradation, keep the platform operational with partially incomplete and inconsistent descriptions

Existing products

There exists a plethora of existing service description efforts that can be grouped into different strands. Each of the strands has its own motivation and representation needs for capturing service information. The individual efforts can be attributed to the following criteria: (i) whether the scope of the effort lies in capturing IT or business aspects of services or the whole service system. (ii) the purpose of the corresponding effort, e.g., enabling of normative data exchange, facilitation of software engineering, or acting as reference model. (iii) Whether the effort is able to capture business network relationships between services. (iv) Whether the effort is standardized.

The first strand of service description efforts is the field of Service-oriented Architectures (SOA). SOA is a way of thinking about IT assets as service components, i.e., functions in a large application are factorized in stand-alone services that can be accessed separately. Because of their IT focus, most approaches limit their attention to the field of software architecture. Originally, several standards bodies specified roughly two dozens of different aspects which are collectively known as WS-* (incl. WSDL, WS-Policy, Ws-Security, etc.). Since one of the key components of a SOA is a service registry, the OASIS standards body introduced the concept of Universal Description, Discovery and Integration (UDDI), i.e., a specification for a platform-independent registry. UDDI services shall be discovered according to information such as address, contact, known identifiers, or industrial categorizations based on standard taxonomies. However, UDDI does hardly prescribe any schema for such information. As the concept of SOA matured, calls for support in software and service engineering increased. Hence, the OMG standards body dedicated its focus to software engineering for SOA, and, subsequently defined the Service-oriented architecture Modeling Language (SoaML). Finally, the multitude of description efforts and different definitions of SOA led to a Reference Model for Service Oriented Architecture (SOA-RM) from OASIS. Similarly, The Open Group drafts an alternative reference model in form of an ontology for Service-Oriented Architectures (SOA Ontology).

A second strand consists mainly of ontologies in the field of Semantic Web Services. The main goal of Semantic Web Services approaches is automation of discovery, composition, and invocation of services in a SOA by ontology reasoners and planning algorithms. The most prominent efforts are OWL-S and WSMO. Many similar efforts have surfaced in literature. With the many approaches around came the need to specify a reference model for semantic SOAs. Consequently, the OASIS is also about to specify a Reference Ontology for Semantic Service Oriented Architectures (RO-SOA).

The third strand is rooted in the rise of on-demand applications that led to the notion of software-as-a-service (SaaS), covering software applications (e.g., CRM on-demand) and business process outsourcing (e.g., gross-to-payroll processing, insurance claims processing) to cloud and platform services. The emphasis of service here implies that the consumer gets the designated functionality he/she requested together with hosting through a pay-per-use model. Thus, software-as-a-service is not synonymous with SOA. This difference triggered the Software-as-a-Service Description Language (SaaS-DL). SaaS-DL builds on WS-* to capture SaaS specificities in order to support model-driven engineering. The strand of SaaS also contains a standard, namely, the W3C recommendation called SML (Service Modelling Language). One anticipated use for SML is to define a consistent way to express how computer networks, applications, servers, and other IT resources are described or modelled so businesses can more easily manage the services that are built on these resources.

The fourth strand is driven by schools of business administration and focuses on capturing the purely economic aspects of services regardless of their nature (with less or no focus on IT services and software architectures). The German standard DIN PAS 1018 essentially prescribes a form for the description of services for tendering. The structure is specified in a non-machine-readable way by introducing mandatory and optional, non-functional attributes specified in natural language, such as, classification, resources, location, etc. The PhD thesis of (O'Sullivan) adopts a wider scope and contributes a domain independent taxonomy that is capable of representing the non-functional properties of conventional, electronic and web services. The work compiles the non-functional properties into a series of 80 conceptual models that are categorized according to availability (both temporal and locative), payment, price, discounts, obligations, rights, penalties, trust, security, and quality.

The fifth strand is also economic but draws attention mainly to describing Service Networks, i.e., the ecosystem and value chain relationships between services of economic value. The e3Service ontology models services from the perspective of the user needs. This offers constructs for service marketing, but in a computational way, such that automated reasoning support can be developed to match consumer needs with IT-services. The main focus of this work is to generate service bundles under the consideration of customer needs. The Service Network Notation (SNN) captures similar aspects to the e³Service ontology. However, SNN is an UML model that can be analyzed for measurements of added value for each single participant as well as for the whole network optimization of value flows.

Finally, there are overarching efforts that concentrate on the bigger picture of service systems or service science also taking into account socio-economic aspects. Stephen Alter was one of the first to realize that the concept of a service system is not well articulated in the service literature. Therefore, he contributes three informal frameworks as a first attempt to define the fundamentals of service systems. The work of Ferrario and Guarino can be seen as a continuation and formalization of Alter's approach. Although differing in its main notions, they present a reference ontology for ontological foundations of service science which is founded on the basic principles of ontological analysis. In turn, this reference ontology forms the core part of the TEXO Service Ontology which extends it by ontology modules for pricing, legal, innovation, or rating information.

USDL is the only effort which covers IT and Business aspects, serves both a reference and exchange purpose, considers business network related information and is about to be standardized.

5.2.2 Repository

5.2.2.1 *Target usage*

The Repository is a place to store service models, especially USDL descriptions but also other models required by components of the overall delivery framework (e.g. technical models for service composition and mashup). The repository provides a common location for storage (centrally or distributed and replicated), reference and/or safety.

The use of a repository is required in order to appear at the marketplace or other tools referring to a number of central repositories for information relevant for interoperation of the enablers and roles within the FI-Ware platform. The repository contains published descriptions which can be utilized by any component in respect to privacy and authorization constraints imposed by the business models. Usually a repository is under control of an authority and usually is keeping track of versions, authenticity and publication dates.

User roles

- The Provider creates services and has an original description describing basic service information as well as technical information. He needs to upload and publish service descriptions on the repository in order to make them available to other components of the platform, such as the Shops/Stores, Aggregators, etc.
- The Aggregator can use for example technical and service-level information for existing in the repository for the purpose creating composite services or mashups from existing services. The Aggregator needs information about the functional and technical interfaces of a service in order to provide an implementation. Service descriptions for the newly created composite service can be uploaded and published to the repository again.
- The Broker needs all kind of business relevant descriptions of services, such as general descriptions, business partners, service-levels, and pricing, to be presented in the shop/store. Also technical information can be required, on a level to be able to do comparisons between services for the consumer.
- The Channel Maker needs detailed information about the channel to ensure the proper channel creation or selection. Further a channel may require embedding or wrapping the service so it can be accessed by the user through the specific channel. Various channels and devices such as Web (browser), Android, iOS but also global as well as local social networking and community platforms such as Facebook, LinkedIn, MySpace, Xing, KWICK! might be supported.
- The Hoster requires information on service-level descriptions, deployment and hosting platform requirements to provide the necessary infrastructure in a reliable and scalable way.
- The Gateway will use information about technical interfaces to provide data, protocol and process mediation services. The gateway also provides services for mediation towards premise systems outside of the FI-Ware platform.

The repository provides also a shared storage for all metadata related to application components, that is, information related to the description of an application component, its associated business model, the user feedback, technical information and other relevant declarative/descriptive information. To make this possible the **USDL** model will be used and extended in order to specify the initial meta-information, both technical and business/market related, empowering the acquisition and integration of new application components from external sources.

The repository will allow managing and sharing all application and services ecosystem relevant information for the whole platform. This includes the provision of relevant semantic information to the FI-WARE Data/Context Management in order to be exploitable by the whole FI-WARE platform to improve and enrich the platform‘s recommendation abilities.

There can be many repositories. A repository can be operated by the service provider (his own web presence), the market place owner as well as any other stakeholder.

Users of the repository have to cope with multiple distributed instances based on different implementation technologies. Therefore the API and format specifications need to be defined clearly.

Hence, the developers, domain experts, and users can use different composition tools interacting with one or multiple repositories to create compositions while at the same time taking into account a multitude of different (business) aspects, such as participating parties, ownership and licenses, pricing, service level constraints, and technical implementation.

5.2.2.2 *GE description*

A suitable API for reading, filtering, and aggregating service information from the repository as well as maintaining service descriptions will be made available for the interaction of other tools and components with the repository and such allowing a tight integration. Figure 45 shows the interaction of some components and tools with the service repository. One important source for service descriptions in the repository is the USDL Authoring tool. In different versions it allows the various stakeholders to create services descriptions or parts of the description in respect to certain aspects. The Authoring Tool can use the Repository API to retrieve, write and publish service descriptions on the repository built-in into the tool. The USDL Crawler can find service descriptions (in its serialized form such as XML/RDF or RDFa) on the Web and import it into the repository by using the writing functionality of the Repository API. A special Web-based Repository Management application for example can be used to organize and maintain information in the repository.

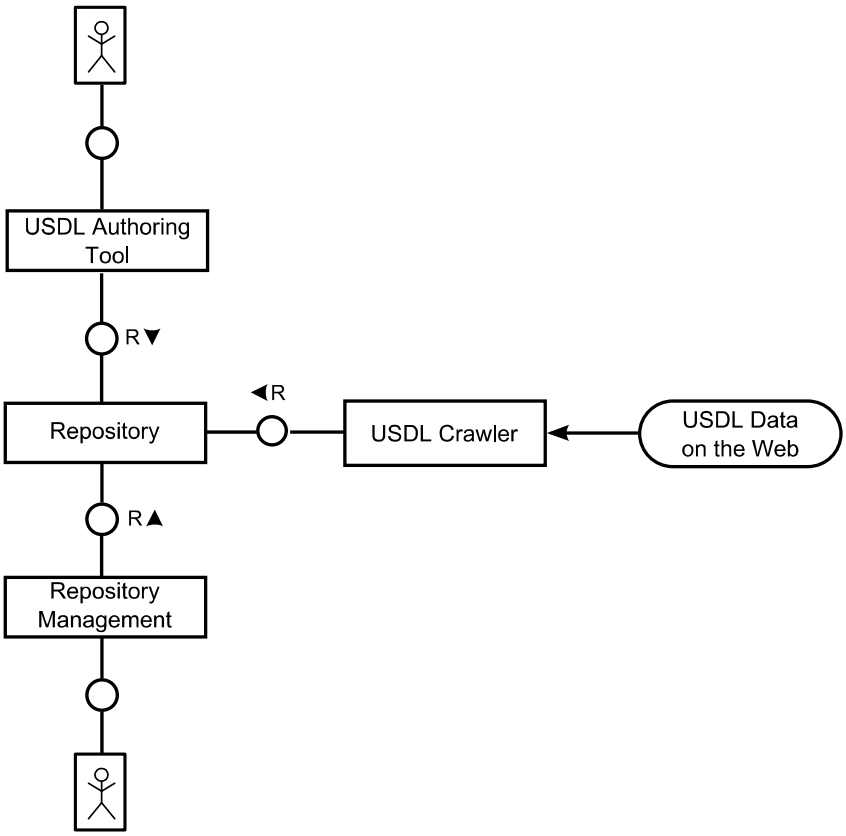


Figure 45 Model Repository for USDL

Other components using the repository are components from the Aggregator, Channel Maker, Broker, Gateway, Provider, and Hoster (see Figure 41).

Functionality

- The repository allows storing service model descriptions such as expressed by USDL.
- Searching and filtering allows to access context specific information.
- Retrieve specific service description elements.
- Import and export of service descriptions for transport from and into other systems.
- Organizational management
- Maintain consistency and constraints within the repository.
- Provide version management and version tracking.
- Authorization and access control realized by the FI-Ware framework services.

5.2.2.3 *Critical product attributes*

- Flexible object model for covering various models.
- Allows for extensions and variants of existing models.
- Scalability: The repository must be able to store huge amount of models.
- Optionally distributed architecture.
- Based on Internet standards
- Easy on-boarding (e.g. through Web-based access and simple registration)
- Web-API for integration into other tools/applications

Existing products

There are various approaches for metadata repositories depending on the underlying information model. Most prominent are the relational data model utilized by relational databases, the XML information model and XML databases like eXist, or RDF graph model and RDF repositories like iServe. However, the databases are only the technical foundation for the model repository. One important issue for a repository of service descriptions is to ensure consistency of metadata. Data stored into the repository and referenced by applications and platform components need it to keep the information consistent in order to ensure reliable operation of the platform. Within previous the projects TEXO and Premium Services various implementations of a USDL repository based on an XML serialization of the USDL eCore model were developed and used. Within FI-Ware we will consider Linked Date representations of USDL in order cope with various extensions and variants of USDL. Semantic metadata repository enablers provided by FI-Ware, which can be the basis of a Linked Data version of the model repository.

5.2.3 Registry

5.2.3.1 *Target usage*

The Registry acts as a universal directory of information used for the maintenance, administration, deployment and retrieval of services. Existing (running) service endpoints as well as information to create an actual service instance and endpoint are registered.

User roles

- Provider uses the registry to discover actual service endpoints at runtime.
- Platform operator provides deployment information for services.
- Hoster updates actual address and access to service endpoints
- Service provider provides deployment options.

5.2.3.2 *GE description*

The Registry links model descriptions (f.i. USDL descriptions in the USDL repository) with technical runtime information. Similar to a UDDI Registry for web services it is a place to find information for technical integration. Only if the service endpoint is registered it can actually be used for service composition and coordination by the FI-Ware platform.

The registry maintains the master data that is needed to ensure proper (inter-)operation of the platform and their components.

Functionality

- Create/read/update/delete entries
- Searching and querying entries
- Management (authorization, logging, …)
- Locating services (find service endpoints)
- Description of deployment according to the technical models

5.2.3.3 *Critical product attributes*

- High scalability to support large number of active services and users.
- High availability to ensure mission-critical real-time business processes.
- Easy on-boarding of new members and services.
- Web-API for integration into other tools/applications

Existing products

Today the UDDI registry fills some of the functionality of the Model Repository but has a limited scope and is not used in a larger Web context. LDAP is used as yellow pages for maintaining organizational data or technical directories.

5.2.4 Marketplace

5.2.4.1 *Target usage*

Internet based business networks require a marketplace and stores, where people can offer and deal with services like goods and finally combine them to value added services. On the marketplace you can quickly find and compare services, which enable you to attend an industry-ecosystem better than before. Services become tradable goods, which can be offered and acquired on internet based marketplaces. Beside automated internet services this also applies for services that are provided by individuals. Partner companies can combine existing services to new services whereby new business models will be incurred and the value added chain is extended.

Given the multitude of apps and services that will be available on the Future Internet, providing efficient and seamless capabilities to locate those services and their providers will become key to establish service and app stores. Besides well-known existing commercial application stores like Apple App Store, Google Android Market, and Nokia Ovi, there are first efforts to establish open service and app marketplaces, e.g. in the U.S. Government's Apps.Gov repository and Computer Associates' Cloud Commons Marketplace. While these marketplaces already contain a considerable number of services, they are currently, at a premature stage, offering little more than a directory service. FI-WARE will fill this gap by defining generic enablers for marketplaces and providing reference implementations for them.

User roles

- Service provider will place offers on the marketplace or in a service/app store.
- Consumer can search, browse and compare offers
- Repository will be used to get services descriptions
- Registry will be used to register stores, providers, marketplaces, …
- Service store will participate on a marketplace and publishes offerings.
- Channel Maker will consumers give access to the marketplace

5.2.4.2 *GE description*

We differentiate the service marketplace from a service store in our architecture. While a store is owned by a store owner who has full control over the specific (limited) service portfolio, and offerings a marketplace is a platform for many stores to place their offerings to a broader audience and consumers to search and compare services and find the store, where to buy. The final business transaction (buying) is done at the store and the whole back office process is handled by the store. There are existing Internet sales platforms that actually have marketplace and store functionality combined. However, conceptually the distinction is useful in order to simplify the architecture and have a better separation of concerns.

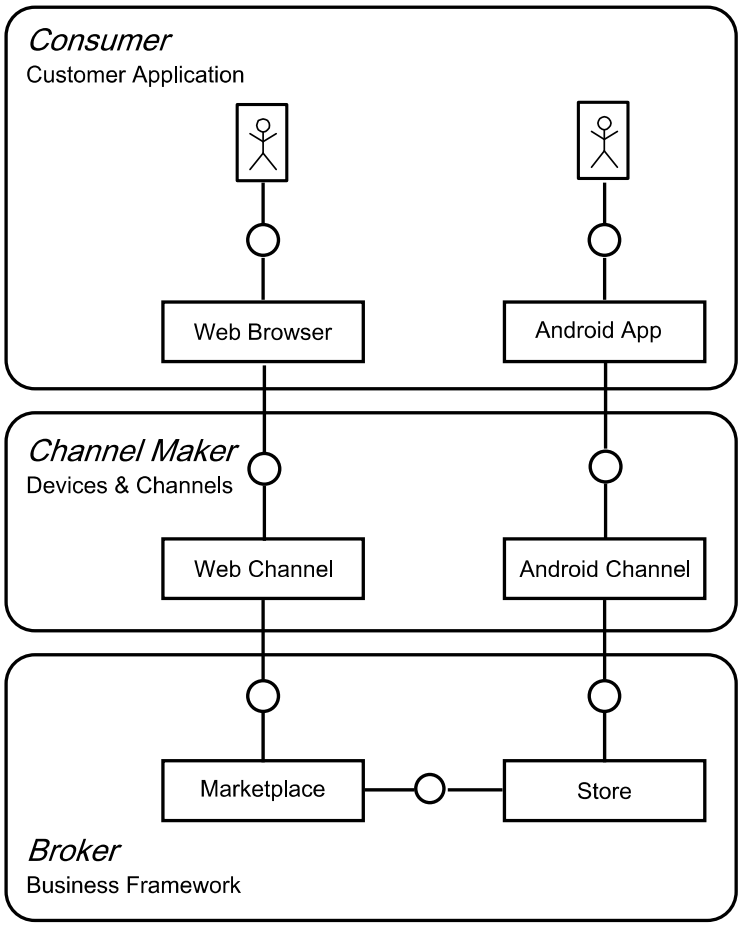


Figure 46 Service Marketplace and Store to Consumer

Figure 46 depicts the interaction of the marketplace and store to bring their services to the consumer via different channels. The marketplace for instance can use a Web channel, which can be used with a standard Web browser, whereas the store is delivering services via a Android device using native applications. The marketplace generic enabler does not have a single user interface. It rather enables to offer marketplace functionality (services) through different channels.

Figure 47 shows the interaction of the marketplace with the repository, registry and store. There might be multiple instances of all components. A marketplace for instance can use multiple repositories and registries as a source and can have a large number of stores offering their services. Both, marketplace and store are using the repository and registry to retrieve and maintain service descriptions.

As a special value added tool for providers and aggregators, a pricing simulator can be offered at the marketplace. The pricing simulator is a decision support system for strategic pricing management. The aim is to support complex pricing decisions that take both inter-temporal and strategic dependencies into consideration by providing a comprehensive market model representation. A tool to tackle complex strategic pricing decisions has to be capable of taking into account the competitive landscape and its development over time. The cornerstone of such a tool is the realization of a stochastic multi-attribute utility model (probably into an agent-based simulation environment), where it can subsequently be fed by either the fitted part-worth of a conjoint study or the relative quality and price scores of a customer value analysis. The result of the tool provides a forecast how different initial price strategies may unfold in the market.

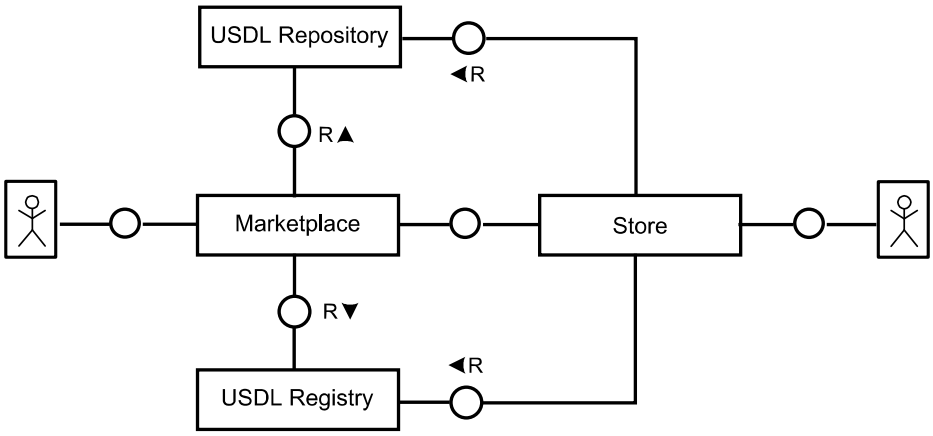


Figure 47 Marketplace and Store to Registry

The functionality listed here contains a number of mandatory features and also a number of nice-to-have features as well. While searching for services, comparing services, and managing connections and interactions with shops are absolutely necessary, the other features are nice-to-have for a marketplace. In the FI-WARE context, request for quotations, ratings, and strategic pricing support seem to offer added value.

Functionality

- Search and browse offers from different service stores.
- Compare offers from different stores.
- Check availability of an offering in the store.
- Request for quotation processing / negotiation for a certain need (optional).
- Independent trustee and clearing house.
- Auction, bidding (optional).
- Advertisement, campaigns (optional).
- Rating, feedback, recommendation of stores and offerings.
- Pricing support, price monitoring, sales cycles in the market across different stores.
- Manage connections and interactions with service shops.

5.2.4.3 *Critical product attributes*

- Customizable for different application domains/sectors.
- Multi-channel access (Web-based access, mobile, …).

- Easy on-boarding for store owners and consumers

Existing products

There is a plethora of marketplaces for various domains. It is useless to give an extended list here. Prominent examples are ebay.com, craigslist, pricefalls and Amazon[20]. In the area of services there are markets for craftsman such as myhammer.de. There are also market places for regional players. Since there is no standard in respect to offerings as well as services and products and no common business or technical framework, it is quite difficult for shop owners to be present on multiple market places. In the area of software applications we find a number of so called App Stores (Apple, Google, and Amazon) that are somehow closed environments controlled by a single owner. The AGORA Service Marketplace was developed within the THESEUS/TEXO project.

5.2.5 Business Models & Elements Provisioning System

5.2.5.1 *Target usage*

The more important question for an application or service when it is available to be launched in the market is to define the business model and the offers and prices available for the customers. The aim is the monetization of those new services and applications. Then it is necessary to have a flexible way to define the manner in which services and applications can be sold and delivered to the final customers; it can be summarized as the business model definition. The business model will define the way in which customers pay by application and services and the way in which the incomes will be split among the parties (single party and multiparty models). Once time the business model is defined, is necessary to provision these details in the rating/charging/billing systems.

User roles

- Managers will setup available business models and the parts of them that will be available for applications, services, parties and users.
- Parties/providers have to setup the business models elements of their applications and services.

5.2.5.2 *GE description*

There is a complex issue to solve when dealing with the aggregation and composition of new services based on other ones that affect to business aspects.

Business models & elements description:

- Offers and price descriptions

[20] Amazon is actually a certain mix of a store and a marketplace. At the one hand Amazon sells products on its own but also lists product offers from external suppliers.

- Policy rules to manage prices
- Promotions description about the offer and prices
- Business SLA (violations and penalties)
- Techniques regarding aggregation, composition, bundling, mash-ups, settlement and revenue sharing business models

Conceptual architecture about BMEPS is shown in Figure 48.

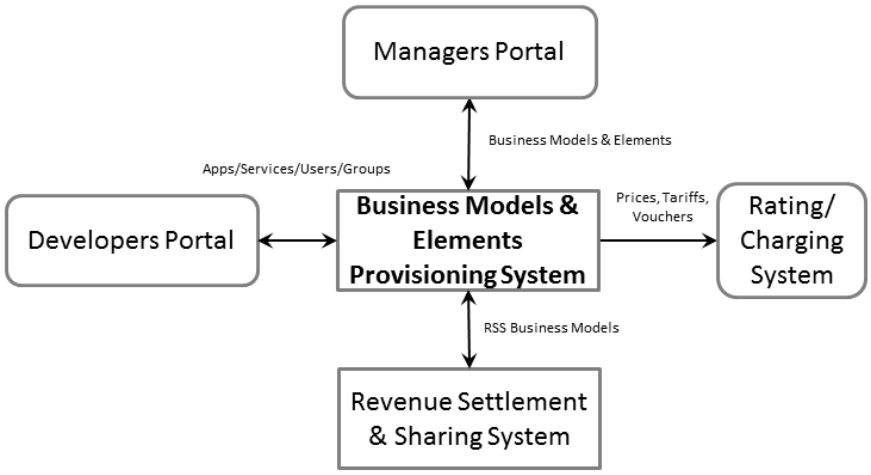


Figure 48 Business Models & Elements Provisioning System

Functionality

- To define all the business models elements to be available for applications and services
- To associate business models elements to applications and services
- To define revenue settlement and sharing models
- To link revenue sharing models to applications, users or/and user groups.
- To provision business models elements in external rating/charging systems and RSSS.

Relations to other components

- Settlement and revenue sharing system
- Developers portal

5.2.5.3 *Critical product attributes*

- There must be possible to support different kind of business models.
- Business models and elements must be customizable.

Existing products

This kind of functionality must exist in an ad-hoc way inside AppStores (Apple, Google, and Amazon) and open APIs from Telco initiatives. It would be attached to rating and billing systems. Inside them it may exists similar tools that may provide this functionality.

5.2.6 Revenue Settlement & Sharing System

5.2.6.1 *Target usage*

In the Future Internet there is a need to manage in a common way how to distribute the revenues produced by a user's charges for the application and services consumed. When a customer buys an application or service, he pays for it. But this charge can be distributed and split among different actors involved (for instance marketplace owner earns money and mash-ups have to split the money).

User roles

- Managers will setup available business models and parts of them are the revenue share model.
- Service provider will setup revenue share model associated to Applications and services, and they have to be loaded in the Revenue Settlement & Sharing System.
- Developers have to know about the revenues of their applications and services.
- Involved service/applications providers have to know about the revenues of their applications and services.

5.2.6.2 *GE description*

In the services oriented environments, there will be a common pattern for services delivery: it does not matter the type of service, each time there will be more composite services based on the aggregation of multiple atomic services. Beyond the complexities of the management of composite services (design, provision, etc.), there is a complex issue to solve when dealing with the business aspects. Both the composite and the atomic services must be accounted, rated and charged according to their business model, and each of the service providers must receive their corresponding payment.

The service composition process will end up in a value network of services (an oriented tree) in which the price model of each service and its share of participation in the overall services is represented. On the other hand, depending on its business model, the business framework may play different roles in relation to the service providers. These realities will lead to different scenarios in which the revenues generated by the services must be settled between the service providers:

- If the business framework charges the user for the composite service, a settlement process must be executed in order to redistribute the incomes as in a clearing house.
- If the business framework charges for all the services in the value network, then besides the settlement function, there could be a revenue sharing process by which a service provider might decide to share a part of its incomes with the service provider that is generating the income.

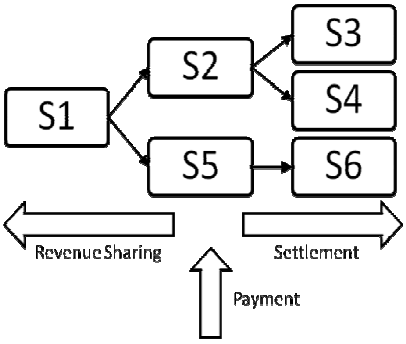


Figure 49 Payments, Settlement and Revenue Sharing in a Services Value Network

In this context, a service must be understood in a broad sense, that is, not only as a remote decoupled execution of some functionality, but also considering other types of services: the business framework itself, content services, advertisement services, etc.

Nowadays, there are some examples in which revenue distribution is needed. The best known example is Apple Application Store[21], which pays a percentage of the incomes form an application download to its developer. Another example is Telco API usage. There are two sides like Telefónica's BlueVia[22] or Orange's Partner[23], in which the application developers receive revenue share for the usage of Telco APIs by the final users. There are also examples of this in the cloud computing services, like dbFlex[24] and Rollbase[25]

The Figure 50 shows a conceptual architecture of a system for settling and sharing revenues. There are a number of different sources of revenues for a given service that will be integrated and processed according to the business model of each service and the revenue sharing policies specified for each partner. The final revenues balance will be transferred to a payment broker to deliver the payments to each provider/developer.

[21] http://store.apple.com

[22] http://www.bluevia.com/

[23] http://www.orangepartner.com

[24] http://www.dbflex.net/

[25] http://www.rollbase.com/

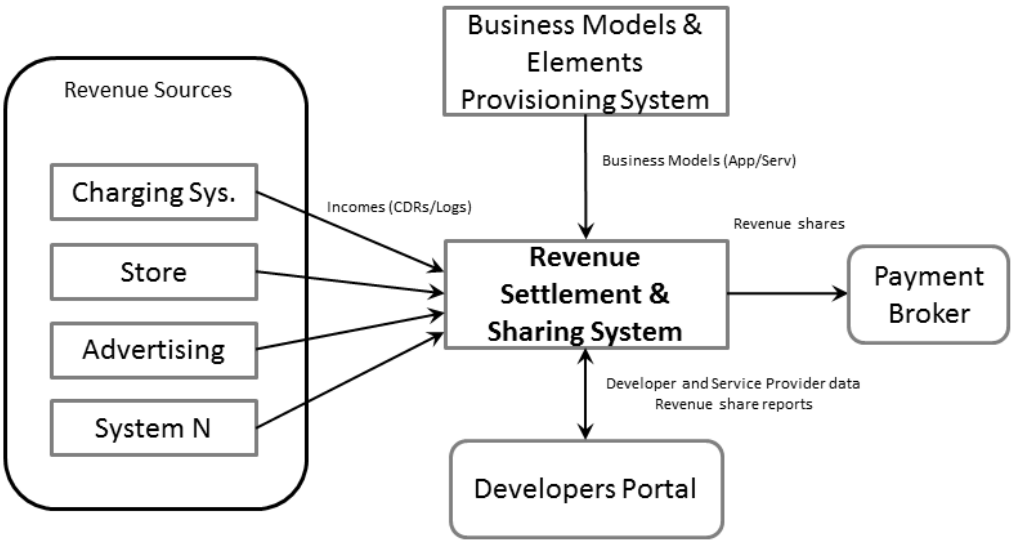


Figure 50 Revenue Settlement & Sharing System

Functionality

- Receive or interact with other external system for loading business models models regarding sharing and settlement.
- Define and store the different revenue sharing models to be applied taking into account Application and Services Ecosystems business models.
- To receive, store and load call data records or charging logs about the different sources of charges of the application and services ecosystem to the customers.
- Creation of aggregated information and data to be used to distribute the revenues.
- The information of developers or users to be paid has to be stored.
- Daily revenue share execution and generation.
- Payment file generation and sending to the payment broker.

Relations to other components

- Business Models Provision System
- Revenue sources systems (application stores, service shops, advertising, etc)
- Payment broker
- Developers portal

5.2.6.3 *Critical product attributes*

- Revenue sharing models must be customizable.
- There must be available simulations of RS models in real time.
- High scalability and high volume of CDRs
- There must be possible to process different revenue sources.
- Report generation API to extract information.

Existing products

There exist various application stores and service ecosystem from different domains. There are well known examples like AppStores (Apple, Google, and Amazon) and BlueVia and Orange Partner initiatives from the Telco world. Inside his commercial products exists similar systems that provides this functionality.

5.2.7 SLA Management

5.2.7.1 *Target usage*

The management of Service Level Agreements (SLAs) will be an essential aspect of service delivery in the future internet. In a competitive service market place, potential customers will not be looking for "a" service, but for "the best" service at the "best price". That is, the quality of services (QoS) – such as their performance, economic and security characteristics - are just as important, in the market place, as their functional properties. Providers who can offer hard QoS guarantees will have the competitive edge over those who promote services as mere 'functional units'. SLAs provide these hard guarantees: they are legally binding contracts which specify not just that the provider will deliver some service, but that this service will also, say, be delivered on time, at a given price, and with money back if the pledge is broken. The cost of this increased quality assurance, however, is increased complexity. A comprehensive and systematic approach to SLA management is required to ensure this complexity is handled effectively, in a cohesive fashion, throughout the SLA life-cycle.

User roles

Specifications will cover the following use cases:

- Service providers design & publish SLA templates as rich descriptions of their service offers.
- Consumers search SLA template repositories for service offers matching their functional & non-functional (QoS, economic, security) requirements.
- Consumers and providers negotiate SLAs.
- Brokers/Hosters (possibly third party) observe the state of service delivery mechanisms in order to detect & report (potential) violations of SLA guarantees.
- Autonomic controllers (managers) respond to changing contingencies (e.g. violation notifications) - by making appropriate modifications to SLA assets and/or service delivery systems - to ensure business value is optimised.

5.2.7.2 *GE description*

SLAs have implications for the entire enterprise context (Figure 51). In particular, SLAs have:

- *Legal Impact*: an SLA represents a binding legal agreement. By entering an agreement, the agreement parties commit themselves to satisfying the terms of the agreement and fulfilling its obligations. To have any significance at all, these obligations must be enforceable – by one means or another - such that failure to abide by the agreement carries real valued penalties.
- *Systems Impact*: providers must ensure they have the means & resources to deliver the agreed functional capabilities within the guaranteed QoS limits. Customers must ensure that restrictions or requirements on service usage are observed. Monitors must ensure that relevant system state properties are observed, and that timely, accurate warnings and/or notifications of violation are posted.
- *Business Impact*: SLAs represent revenue, investment and risk. Customers pay for the services they consume. Providers pay for the resources they exploit to deliver services. SLA guarantee violations incur penalties. The goal of SLA management is to manage SLA assets in order to optimise business value.

A prerequisite of effective SLA management is the systematic integration of knowledge at all these levels.

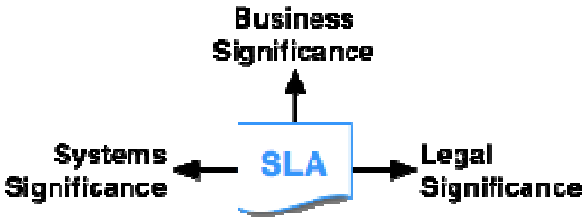


Figure 51 The impact of SLAs on the enterprise context

The SLA life-cycle (Figure 52) begins with an understanding of the service provider's business objectives & models and their relation to available resources and service delivery capabilities. This combined knowledge informs the *design* of the provider's service offer, encoded and published in the form of an SLA "template" (an SLA with open, customisable "slots" for customer specific information) to support enriched, QoS-based service *discovery*. Having located a suitable or merely promising template, the customer initiates *negotiation* with the provider, which proceeds in rounds of SLA proposals and counter-proposals until agreement is reached. If agreement is reached, the SLA is signed; resources allocated, and service delivery and (possibly third party) *monitoring* mechanisms commissioned. Once in effect, the SLA guarantees must be monitored for violation, and any penalties paid. If possible, steps may also be taken to assure optimal business value: service delivery and monitoring systems can be reconfigured (*cf.* internal service level management) and SLAs renegotiated or even terminated. For the future internet, we expect - and so wish to support – increasingly *autonomic control* of negotiation, service-delivery and monitoring. Finally, there are various management issues (not indicated in Figure 52) relating to the *versioning* of templates (offers) and *archiving* of SLAs, monitored data, SLA state & negotiation histories, and any other information that may prove useful to the design of future service offers.

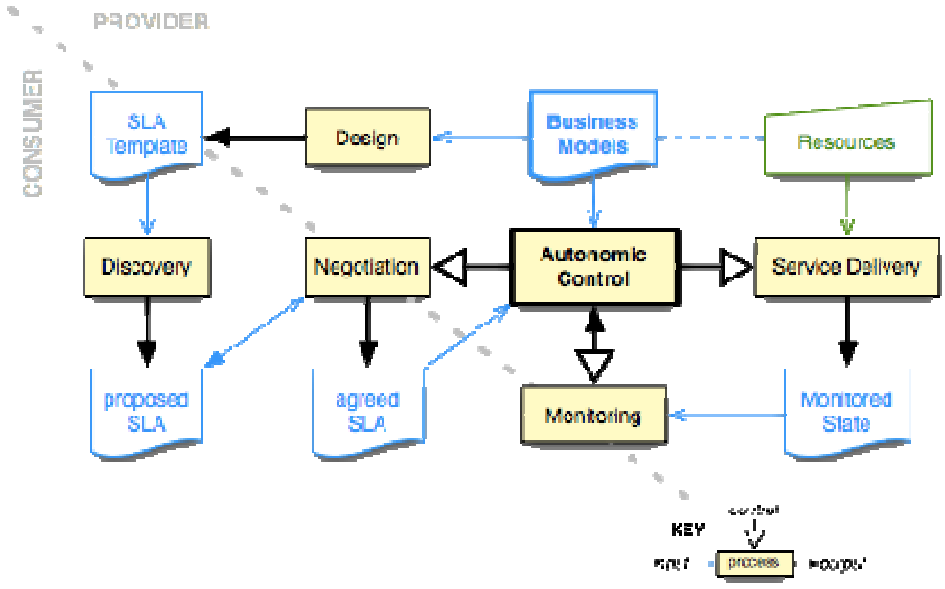


Figure 52 Process view of the SLA life-cycle

To reiterate, the key point is that management of the SLA life-cycle critically, and fundamentally, depends on understanding the *significance* of SLAs to the whole enterprise. SLAs have *legal* impact, *systems* impact and *business* impact. SLA management is all about controlling this impact, throughout the entire SLA life-cycle, in order to optimise business value. To tackle this highly complex problem space, we first need to understand it. Current standards and technologies offer at best only partial solutions: tackling either one aspect in isolation, or looking at the whole but with overly restrictive assumptions. What is missing is the big picture: *a comprehensive and highly integrated set of generic information & process models detailing SLA management over the entire SLA life-cycle*. The FI-WARE generic enabler for SLA Management will look to develop this integrated view.

Specifically, the generic enabler will consist of a comprehensive "SLA Model" comprising three mutually specified sub-models (Figure 53):

- *SLA content model*: formal conceptual, syntactic and semantic specifications of SLA content. In particular, SLA content must be *clear and precise*. It should be possible to get a clear indication of the terms of the SLA from only a superficial reading. But these same terms must also have a precise significance at the technical systems level – and in particular they must translate to unambiguous monitoring requirements.
- *SLA life-cycle model*: formal specifications of SLA state-transitions and state semantics. For example, the precise conditions under which an SLA can be formally described as "agreed", "terminated" or "violated".
- *SLA management model*: formal specifications of processes & mechanisms impacting SLA state. – e.g. functional capabilities & abstract machines.

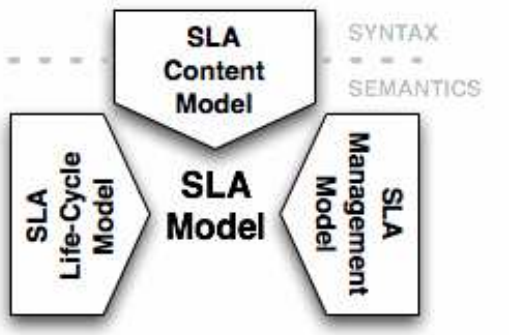


Figure 53 The SLA Model

The SLA Model will not be designed from scratch, but will instead be consolidated from existing solutions. In particular, we will look to extend and generalize work undertaken as part of the FP7 ICT Integrated Project SLA@SOI[26], and to consolidate these results with other notable efforts such as: WS-Agreement[27] (*re:* negotiation), WSLA[28] (*re:* content & monitoring) and SLAng[29] (*re*: monitoring).

[26] SLA@SOI, see: http://sla-at-soi.eu

In terms of the high-level business framework component architecture, there is no need for a dedicated SLA Manager component. All the SLA management processes described above can be viewed as either advanced functions of existing components or as external service dependencies: *discovery* is properly the province of the Marketplace; *negotiation* the province of the Shop; *monitoring* is ideally left to independent, trusted, third-parties; and the remaining information requirements are covered by USDL and Business Elements & Model Provisioning. This said, existing tools and services (e.g. from SLA@SOI) will be employed if and where applicable, and proof-of-concept and/or prototype demonstrators will be constructed for the more advanced features of SLA management. The goal of the FI-WARE generic enabler for SLA Management, however, is to develop an integrated SLA Model that can serve as the foundation for the development of robust SLA-aware applications in the future internet.

Functionality

Specifications supporting:

- SLA content authoring.
- SLA life-cycle management.
- SLA-based service level management (SLM).

Relations to other components

- USDL: e.g. for SLM support
- Business Models and Elements: for encoding business value
- Revenue sharing
- External (third party) services: e.g. for third-party monitoring and secure penalty payments.
- Marketplace: for QoS-based discovery mechanisms
- Shop: for SLA negotiation.

5.2.7.3 *Critical product attributes*

- Extensibility and customizability to meet (unforeseen) domain-specific requirements.
- Clarity and precision of SLA content: the significance of SLA guarantees at system, business & legal levels must be immediate & intuitive and technically unambiguous.
- Harmonised, comprehensive and integrated SLA Model specifications.
- Modular and extensible design supporting custom application to (unforeseen) domain-specific requirements.

---

[27] WS-Agreement, see: http://forge.gridforum.org/projects/graap-wg

[28] WSLA, see: http://www.research.ibm.com/wsla/

[29] SLAng, see: http://uclslang.sourceforge.net/index.php

Existing products

- SLA@SOI
- WS-Agreement
- WSLA
- SLAng

5.3 Generic Enablers for Composition and Mashup

The recent social, economic, and technological developments lead to a new phenomenon often called servification, which is supposed to become the dominating principle in the economy of the future. Wikipedia, Amazon, YouTube, Apple AppStore, Facebook and many others show the unprecedented success of Internet-based platforms in many areas including knowledge and content delivery, social networking, and services and apps marketplaces. FI-WARE is supposed to play a key role as the main technological driver bringing together cloud computing, mobile networks, Web2.0, Apps, services, data sources, and things on a broadband Internet and enabling multi-channel consumption and mobile multi-device access. **Application and Services Ecosystems** able to exploit the innovative value proposition of servification to its full potential from the technology as well as from the business perspective are envisioned as one of the main pillars of the Future Internet.

Few applications can really become killer applications alone, but many of them could have better chances in combination with others. **Support of cross-selling through composition** would therefore become a highly desirable feature in Application and Services ecosystems. However, most relevant ecosystems today do not incorporate these features or do not incorporate them at the right level. FI-WARE strives to exploit the composable nature of the application and services technologies in order to support cross-selling and achieve the derived network scaling effects in multiple ways. It will enable composition either from the front-end perspective – **mash-ups** or the back-end perspective – **composite services**.

Phenomena like Wikipedia and YouTube have taught us how end consumers may become major drivers of innovation whenever suitable authoring tools, complemented by social tools that maximize their ability to exchange knowledge and gain recognition, are provided. However, while **crowd-sourcing and social web technologies** have experienced a relevant development in the area of information and multimedia content, they are still immature in the application and services space. In FI-WARE, the mash-up and composition capabilities offered by the different types of supported components are expected to leverage their reusability as well as the creation of value-added apps/services not only by application and service providers but also by intermediaries and end users acting as composers or **prosumers**. The supported framework will rely on a defined set of user, provider and intermediary roles defining the skills, capabilities and responsibilities of the actors and relationships among them. The value network spanned by the roles of the actors and relationships among them defines the creation and distribution of the value-added applications from the technical perspective. As the capabilities and skills of actors are expected to range from technical experts with programming skills to domain experts without technical expertise or even simple end-users with no programming or technical skills, all kinds of usability aspects, conceptual simplification, recommendation, autonomous provisioning (incl. composition, description and deployment), as well as procurement and user guidance will be taken into consideration.

There are two main functional components that can be identified in service composition and application mashups: the aggregator and the mediator roles (Figure 54). The aggregator allows the creation, exposition and execution of composed (or mashed up) services and applications. Whenever a composed service or application is used the need might arise to use a mediator for components to properly communicate and interact.

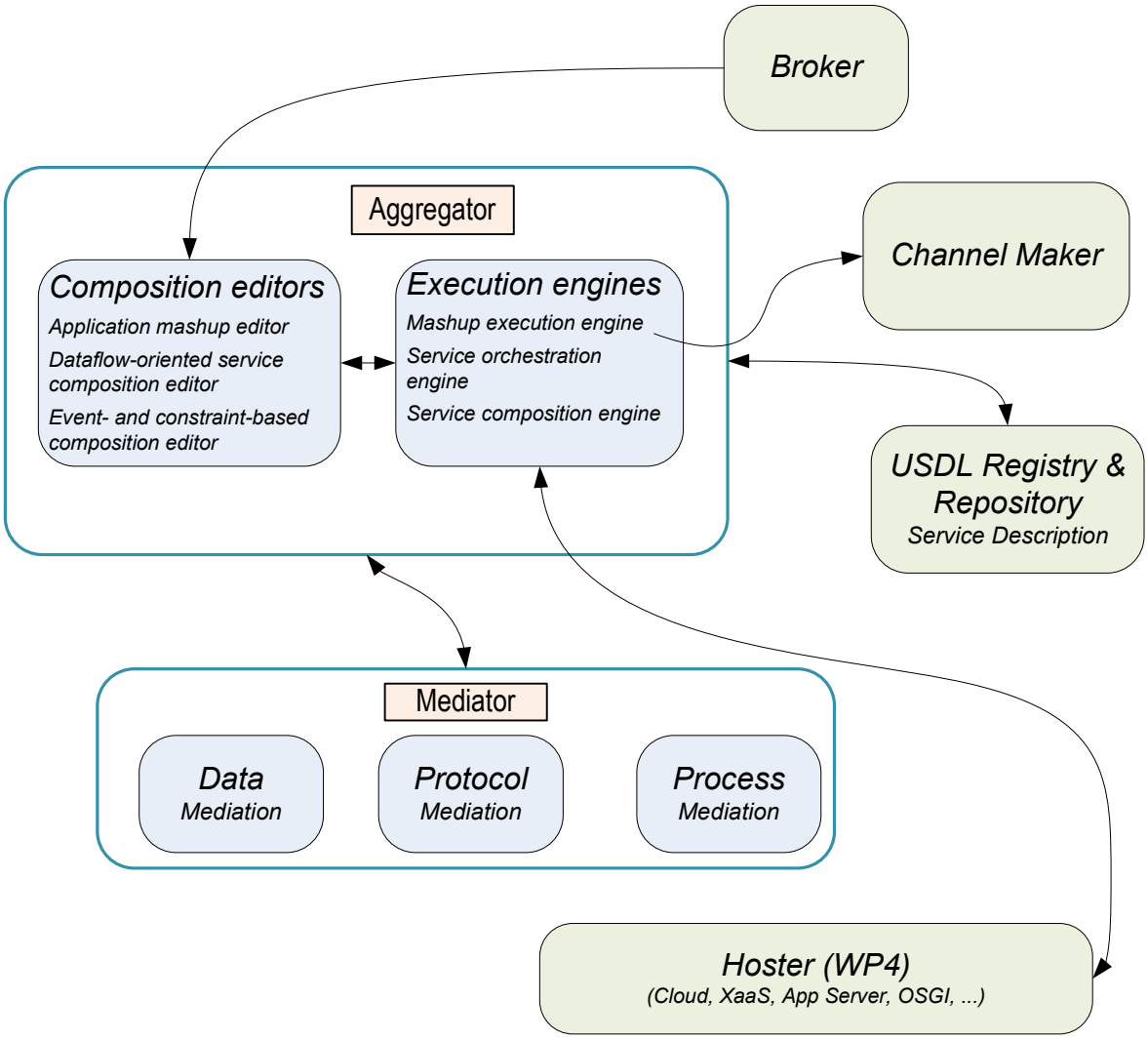


Figure 54 High-level architecture of Aggregator and Mediator

Future convergent composition techniques require a smart integration of process know how, heterogeneous data sources, management of things, communication services, context information, social elements and business aspects. Communication services are event, rather than process driven by nature. Thus, a composition paradigm for the Future Internet needs to enable composition of business logic also driven by asynchronous events. The framework will support the run-time selection of Mashable Application Components (MACs) and the creation/modification of orchestration workflows based on composition logic defined at design-time, to adapt to the state of the communication and the environment at run-time. The integration of Things into the composition requires that the special characteristic of IoT services are taken into account, like lower granularity, locality of execution, quality of information aspects etc. Moreover, the framework allows the transparent usage of applications over many networks and protocols (HTTP/REST, Web Services, SIP/IMS) and corresponding execution platforms (both Web and native apps) via a multi-protocol/multi-device layer that adapts communication and presentation functionality.

The aggregator can be further split into a ***composition editing*** tool used for the creation of design time compositions (described by a specific composition language), an ***execution environment*** to expose and execute the composed services, and a ***repository*** for keeping the relevant information in the meanwhile. Before presenting the functional components of an aggregator we make a short presentation of relevant concepts and technologies.

Front-end vs. back-end composition

Roughly speaking an application represents a software construct that exposes a set of functions directly to the consumer via a user interface, without exposing any functionality to other software constructs. A service on the other side is a software construct that provides functions for other services or applications is accessible via a set of well defined programming interfaces.

From the compositional perspective we can differentiate two broad categories: service composition or application mashup. The two represent composition from either the front-end perspective – composed applications (or mashups), or the back-end perspective – composite services. The main difference is the interaction with the human end-user. In the case of back-end composition, the composed service is yet another back-end service, the end-user being oblivious to the composition process. In the case of front-end composition, every component will interface both other components and the end-user through some kind of user interface. Thus the front-end composition (or mashup) will have direct influence on the application look and feel; every component will add a new user interaction feature. This of course will heavily influence the functionality of the components. While back-end components are created by atomizing information processing, front-end components (referred also as widgets or gadgets) are created by atomizing information presentation and user interaction. Another difference is that the creation and execution of the front-end components will heavily depend on the available runtime environment (e.g. web-browser, device OS, 2/3D presentation platforms), and the different presentation channels they are exposed through (Facebook, Yahoo Widgets).

Composition vs. mashup

The capabilities and skills of composite service creators are expected to range from technical experts with programming skills to domain experts without technical expertise or even simple end-users with no programming or technical skills. Actually one of the main advantages of a component-based architecture is the democratization of the service creation process. You don't need to be a technical expert and know how a component is built in order to use it. Of course the expressivity of the composition language will determine the amount of technical knowledge required to use it. A smart composition editor could cater for different user expertise and roles (from service creators, to resellers and finally to prosumers) by hiding complexity behind different types construction blocs, trading off flexibility for simplicity. For example, a technical expert could write the composition in a text-based format with full access to the constructs of the composition language, while an end-user could use a graphical building block construction employing only the most basic features of the language. Nevertheless the complexity and expressivity of the composition language could be pragmatically restricted by the application area or by the support offered in the execution environment.

Due to the reasons previously explained we expect back-end compositions to employ a more complex composition language and environment (notwithstanding the fact that users such as domain experts with no technical background may use only simple composition language subsets). Front-end compositions on the other hand will most likely use simple constructs and be suitable for manipulation by end users and presentation designers, and more expressively referred as mashups. In the next sections we describe common characteristics for both the front-end mashup and the back-end composition.

Data vs. service composition

Another differentiation that is sometimes made regards data vs. service composition. Data composition is used to denote a composition process where only data flows are transformed and aggregated (e.g. Yahoo Pipes, JackBe), in contrast to service composition, which entails more complex dataflow, workflow, and synchronisation mechanisms (e.g. BPEL). Nevertheless we can regard "service composition" as being a superset of "data composition".

Workflow-based composition

Composite services consist of a set of component services and definition of the control and data flow among them. The actual exchange between elements in a composition can be understood as a workflow with specific actors (the services) and the flow of actions (e.g. a data dependency - one operation needs data produced by another operation) to be executed by them towards achieving the goals specified. Typically such a composition and the related workflow has to be created before the composite service can be executed and thereafter created compositions and workflows will be executed unchanged repeatedly in the same or in different contexts.

Composition creation is the first logical phase in the service composition process. Composition creation functionality aims to support the creation phase by enabling the automated checking of dependencies between services, so that created compositions are valid and useful. General practice in workflow definition and description languages is the definition of fallback services to compensate for faulty execution. Such fallback services can clearly only be defined during the creation phase.

The explicit representation of the execution order of the composition steps in a workflow provides a clear and easy way to express chronological control dependencies. Typically, workflows can express sequential and parallel flows as well as use conditional statements. More advanced approaches can support multiple entry points, error and exceptions handling, invocations of external services. Most often, workflows are defined implicitly by implementations of applications written using typical imperative programming languages. To overcome the aforementioned disadvantages of the implicit workflows and to offer a more formal and programming language independent way of expressing the workflows several workflow definition languages were proposed and standardized (BPEL4WS and BPMN 2.0 are technologies of choice for XML Web Services based workflows). Workflow scripts are executed by an orchestration engine (e.g. a BPEL execution engine executes BPEL4WS workflows). The orchestration denomination represents an analogy to the composition. While “composition” is employed at creation time, the “orchestration” is supervising the execution.

Event-based dynamic composition

Alternatively, workflows can be dynamically created or adapted during composition execution. Suitable services are identified and executed as needed based on the current context (depending on external and internal events and results from previous service invocations).

Such workflows are valid only during execution time and under the particular circumstances. The dynamic creation of compositions can be achieved through systems with the capacity to solve problems using inference procedures with a traceable line of reasoning based on application domain specific knowledge (e.g. model-driven systems). In contrast to rules-based and case-based systems that rely on experience and observations gathered by human users, model-driven systems rely exclusively on formalised knowledge stored within the system itself.

Model-driven systems are enhanced by modelling the constraints that influence the behaviour and functionality of a system and its components. Constraints are constructs connecting two unknown or variable components and their respective attributes, defines the values the variables are allowed to have, and defines the relationship between the two values. In other words, constraints can ensure that specific components are put together in a correct fashion without having to specify any component related rules or calculations.

The model-driven template-based approach to composition creation is based on composition templates (composition skeletons). The skeleton includes the main parts of the business logic of the composed service, but is not complete with regard to some implementation parts. For example, certain implementation parts should invoke some kind of services (SIP, WS, EJB, etc.) or should require some data from a particular source, but neither concrete service nor data source are known at the moment of service development. Instead, such points in a template are marked in a special way, so that this place can be found at the runtime.

During the run-time (and/or even at the assembly or deployment time), the composition engine is invoked at those places and it dynamically decides about what services to invoke or which data source to use based on constraints evaluated at that particular time. Essentially the composition engine is creating the workflow step-by-step during runtime, and different composition decisions can be taken depending on external events or on the return values of previously executed services. It should be noted, that the selected service can itself be another skeleton build in the same way.

As mentioned above, the composition engine creates workflows on the fly at runtime. Ideally it should not deal with the protocol implementation of the invocation of different service technologies (e.g. SIP, WS, REST, RMI, etc.). These are left to the Composition Execution Agents (CEAs), which are responsible for enforcing composition decisions in a technology and protocol specific way. Actually the CEAs are orchestration engines for the different service technologies and they receive the workflow from the composition engine step-by-step via a uniform API. Note that there could be considerable differences between these CEAs. For example WS entails request-response invocations in a hierarchical tree structure while SIP deals with asynchronous events and persistent services in a chain structure.

5.3.1 Composition editors

5.3.1.1 *Target usage*

The *Composition editors* are *Generic Enablers* that help the service provider to create application mashups and composed services. Editors should provide an environment to combine and configure applications and services in graphical way.

Different editors could cater for different user expertise (from technical experts with skilled in the composition language to domain experts without technical expertise or even simple end-users with no programming or technical skills) and roles (from composed service creators, to resellers and finally to prosumers) by hiding complexity behind different types of construction blocs, trading off flexibility for simplicity. By prosumer we denote a consumer-side end-user who cannot find a service which fits her needs and therefore modifies/creates services in an ad-hoc manner for her own consumption.

Editors should support facilities for consistency checking of interfaces and simple debugging. They should connect to the Execution Engines allow testing, debugging, installing, executing, controlling and post execution analysis of the composed applications.

Composition descriptions and technical service descriptions should be edited/created in the editor and stored/fetched to/from the Repository.

When creating compositions/mashups editors might connect o the business infrastructure:

- Marketplace to search for services
- Shops to purchase component services them for testing /deployment, and to expose composed services for purchase
- USDL Registry to browse business information related to services

Editors could be connected to a user and identity management service for controlling access to the applications.

5.3.1.2 *Descriptions of GEs*

As presented in Figure 54 we have identified three different types of GEs that pertain to specific aggregation needs. These are the Application mashup editor, the Dataflow-oriented service composition editor and the Event- and constraint-based composition editor, which are detailed next.

Application mashup editor

Regarding application mashup, FI-WARE reference architecture should offer an editor to create applications built from discrete front-end components (e.g. gadgets/widgets, apps) and connected at the front-end layer. These components rely on a series of either plain or composed backend data/services. For testing and debugging and presentation of logged runs the editor will connect to the Mashup Execution Engine.

The editor should offer functionality for creating an application front-end as a mashup built from gadgets/widgets that rely on a series of either plain or composed backend services. Client-side inter-gadget communication facilities should be supported including support for optional filters (wiring). Design-time semi-assisted modelling aids, such as suggestions on relevant relationships amongst gadgets and mashups (e.g. consumed data vs. produced data) together with dynamic discovery facilities from the gadget/widget and mashup repository will ease the mashup creator's work. For persistence/sharing purposes the mashup needs to generate a MDL (mashup description language) model.

The application consumer, acting as a prosumer (being her domain experts, business professionals or end users), uses the application mashup component within the composition editor to develop mashups. The application mashup provider develops their own valuable gadgets or even mashups and offers them as building blocks. These gadgets/mashups are added to and provided via the repository.

Dataflow-oriented service composition editor

We characterize an editor for dataflow oriented compositions intended to support subject matter expert without programming competence and additional separated features design-time composition. Service providers of different roles (subject matter experts, business professionals or end user prosumers) use the editor to describe and operate services. The services are added to the repository and provided via the library. Operator of the application composer manages and controls application composer access and functionality.

The dataflow oriented application composer is modularized, with major functionalities described next. Composition studio: This module provides a graphical user interface to combine appropriate services, connect services and data flows, configure services and check consistency of the composition. It supports file transactions (open, store, rename) and display options (e.g. expand, link types, descriptions). Debug/Simulation: This module allows the step by step application execution. It supports to display a data flow and the change of the data for every single service. Service Library: This module provides an interface to the repository to browse information about the service categories, services, description, data input and data output. Deployment and governance: This module allows configuring, deploying and managing an application. Aspects to be controlled are start and end time for service execution, authorized users and user groups to execute the application, remove and rename of an application. The deployment feature supports composition model translation into executable languages such as BPEL. The execution is supported by the service orchestration engine (especially BPEL features).

Important features are: (a) Extensions for design-time service composition that can be provided in separate libraries, (b) Modelling workflow, (c) Supporting complex behavioural patterns, using modelling elements such as gateways (exclusive, parallel, loops, etc), (d) Modelling data flow, including support for complex data flow mapping, transformation and consistency check, (e) Modelling of orthogonal (independent) composition work and data flow, (f) Design-time semi-assisted (in opposition to manual modelling) modelling aids, such as dynamic binding, data flow generation, data flow mapping, data flow consistency, (g) Task expansion with matching composition (sub-processes).

Event- and constraint-based composition editor

Next we describe the functionality of an editor that supports convergent composition techniques that include asynchronous event-driven communication services in a SOA manner.

The editor allows the creation of composed service skeletons. The skeletons provide the business logic, the data and control flow, and service placeholders. While looking similar to workflows, they are not, as the

skeletons only provide placeholders for services that are going to be resolved at runtime. Moreover they may not provide a clear ordering of service execution (explicit parallelism). The order can be chosen by the execution engine at runtime depending on the specified constraints that trigger the execution (data availability, external events, etc.).

Specification of global (valid throughout the composition) and local (valid only on that template) constraints are used to decide runtime service selection and event filtering. While the choice of relevant services can differ at runtime compared to design time, still for many cases the set available at runtime is a subset of the one available at design time. Thus the editor should apply smart constraint resolution also at design time to help the designer get an idea of what services might resolve at runtime, an help her prepare all the relevant inputs and outputs. This includes smart automatic editing suggestions to the user (i.e. tab-completion).

Many communication-type services depend heavily on events, and first class support for events needs to be provided. External and internal events may start actions, events can be filtered, events can be thrown, and scopes can be defined on parts of the skeletons and subsequently used in event filtering.

Several services might be suitable to implement a placeholder template. These services can have different input and output parameters and the editor needs to offer the possibility to correctly map the different dataflow types, while providing an easy way to use the unified interface.

5.3.1.3 *Critical product attributes*

Application mashup editor

- Visually compose (mashup), configure, deploy, and test composite (mashup-based) applications in an easy-to-use environment.
- Support for innovation at the service front-end, adapting it to their actual necessities.
- Availability of a Web2.0-based (crowd-sourced) shared catalogue of combinable gadgets/widgets, mashups and services. Find easy-to-understand building blocks, descriptions and examples.
- Easy configuration, modification and arrangement of gadgets/widgets and mashups.
- Rely on Web standards including standards on mashup description languages.
- Openness and extensibility
- Ability of integration in multiple channels (e.g. social network, portal, widget)

Dataflow-oriented service composition editor

- Compose, configure, deploy, and test applications in easy- to-use environment for tech-savvy end users without programming competencies.
- Easy-to-understand descriptions of atomic services and discovery in repository.
- Easy configuration, modification and arrangement of services and applications.
- Providing easy-to-use control and monitoring features of application execution and management.
- Using open web standards for technologies. Thus supports extensibility to integrate other standardised (REST, SOAP interface) services.

Event- and constraint-based composition editor

- Use Web and Web Service standards and SOA and EDA principles.
- Ability to interface many service technologies, especially communication-type services.
- Designer should find easy-to-understand descriptions and examples of atomic and composed services.
- Support for asynchronous event-driven services.

- Support for constraint-based composition with late binding.
- Compose, configure, deploy, and test composed applications in easy-to-use environment.

Existing products

Regarding mashup and gadget specification, there is a draft widgets specification published by the W3C. Software vendors (like Microsoft or Google) defined their own widget model. Mashup platforms such as NetVibes use the compelling Universal Widget Architecture (UWA), whilst others, such as OpenAjax, have no component model per se but vital strategies for fitting/putting Web components together in the same mashup.

There are a large number of FLOSS and Commercial (including free or community editions) service composition editors for BPMN, BPEL, etc, such as Oryx, Intalio, ActiveBPEL, Eclipse BPEL, Eclipse BPMN, JBoss jBPM, Activi BPMN Eclipse Plugin, Oracle BPM Studio.

An event-and constrained-based editor is implemented by the Ericsson Composition Editor.

5.3.2 Composition execution engines

5.3.2.1 *Target usage*

The Execution engine is exposing and executing the composed services. The service provider/operator deploys services/mashups by fetching technical service descriptions and composition description from the repository. This most likely will happen through a graphical user interface in the Composition Editor. The service provider/operator controls execution modes (start, stop, debug), and can fetch logs and tracing data, most likely through the Composition Editor GUI.

5.3.2.2 *Descriptions of GEs*

We can differentiate three generic enablers for execution engines presented next: front-end mashup execution engines, service orchestration engines, and event-based late-binding composition engines.

Mashup execution engine

The FI-WARE reference architecture should offer a mashup container able to execute applications built from discrete front-end components (e.g. gadgets/widgets, apps) and connected at the front-end layer. At an architectural level the concept of mashup container relying on a well-defined platform API vertebrates the reference architecture. This API will offer inter-gadget communication and mashup state persistence facilities. The decentralized nature of mashups demands the Mashup execution engine to coordinate gadget execution and communication within the mashup. The availability of a standardized mashup description language will help decoupling the mashup engine from the registry and repository.

The functionality should ensure coordination of gadget execution and communication within the mashup, creating the communication channels (data flow) between gadgets. It should also handle deployment and execution of mashups, and guaranteeing the mashup state persistence, and finally generating an executable mashup from a MDL (Mashup Description Language) model

Service orchestration engine

Orchestration describes the automated arrangement, coordination, and management of complex services. Orchestration provides an executable business process, where multiple internal and external web services can be combined. The process flow is controlled in the execution environment. WS-BPEL (Web Service

Business Process Execution Language) and BPMN 2.0 (Business Process Modelling Notation) are examples for languages for the orchestration of web services.

Orchestrations based on WS-BPEL and BPMN 2.0 languages have a) facilities to enable sending and receiving messages, b) a property-based message correlation mechanism, c) XML and WSDL typed variables, d) an extensible language plug-in model to allow writing expressions and queries in multiple languages (BPEL and BPMN 2.0 supports XPath 1.0 by default), e) structured programming constructs including if-then-else, if-else, while, sequence (to enable executing commands in order) and flow (to enable executing commands in parallel), f) a scoping system to allow the encapsulation of logic with local variables, fault-handlers, compensation-handlers, g) serialized scopes to control concurrent access to variables.

Alternatively, an orchestration engine could execute sequential workflows steps delivered from a composition engine through a uniform API. Different engines would execute only steps suitable to the protocol/technology implemented (e.g. : SIP, WS, REST, WARP, RMI, JBI).

Common functionality includes a) configuration and enforcement of runtime behaviour of a process using standard policies, b) performing server-based runtime message correlation and handling service communication retries, c) endpoint management to make it easy to deploy an orchestration from one environment to another, or deal with a change in topology, d) suspension of a running process using process exception management capabilities to handle bad data which would otherwise have unnecessarily failed a transaction, and e) a management console to monitor server activity and set performance thresholds for notification.

Service composition engine

For the dynamic late-binding composition, the composition engine creates a workflow on the fly from a matching skeleton. The process is triggered by a composition execution agent (CEA) that receives a triggering event and requests the next step from the composition engine. Based on what the triggering events was, the composition engine selects the matching skeleton and creates a new session. Then, at each step it selects a suitable service that matches all the global and local constraints and serves it to the agent to execute. Execution results from the previous steps together with potential external events can influence the constraint-based decision process selecting the service for the new step. If several services are suitable to implement a certain step one of them is chosen. If a component service fails during execution, the next compatible one might be executed instead.

The engine starts by fetching service description and composition description (skeletons) from USDL Repository, and then it executes the business logic and manages the dataflow transformation and the control flow elements specified in the skeleton step-by-step. At each step it chooses the relevant services for the skeleton execution by applying constraint resolution on context information. The composition engine uses orchestration engine(s) to deliver the on-the-fly created workflow step-by-step via a uniform API. It is also responsible for maintaining an up-to-date structured shared state across sessions, and provides execution traces for debugging and statistics.

5.3.2.3 *Critical product attributes*

Mashup execution engine

- Rely on Web standards including standards on mashup description languages.
- Openness, Extensibility
- Ability of integration in multiple channels (e.g. social network, portal, widget)
- Users executing their mashups from their favourite browser.
- Persistence of the state of the mashup.

Service orchestration engine

- High scalability
- High availability
- High configurability
- High robustness

Service composition engine

- Flexible and robust service composition (including event-based semantic, late binding of services, and constraint resolution).
- Ability to integrate multiple composition execution agents (CEA) orchestrating different protocols via a common API.
- High scalability and high performance

Existing products

Regarding mashup execution engines, there are a plethora of products, including EzWeb, Jack Be, Google IG, Yahoo! Pipes, NetVives, Open Kapow. Each of them is tackling the problem with a different approach.

Orchestration engines examples are as following. BPEL: IBM WebSphere Business Integration Server Foundation, Oracle: BPEL-PM, CapeClear. Open-source implementations: Apache ODE, OW2 Bonita, ActiveBPEL, Bexee, PXE BPEL. BPMN 2.0, JBoss jBPM: Alfresco Activiti, and Ericsson Composition Execution Agents: SIP, WS, REST, WARP, RMI, and JBI.

The Ericsson Composition Engine implements a dynamic event- and constraint-based execution engine.

5.4 Generic Enablers for Gateway

5.4.1 Mediation

5.4.1.1 *Target usage*

Providing interoperability solutions is the main functionality of the mediation functionality. The heterogeneity that exists among the ways to represent data (i.e. to represent syntactically and semantically the information items that are requested or provided by an application or a service), and to represent the communication pattern or the public process needed to request a functionality (executing a composition in a different execution environment or implementing dynamic run-time changes might require a process mediation function), are problems that arises as soon as a services has been dynamically discovered at run-time. Acknowledging the necessity to deal with these heterogeneities, dynamic mediation solutions are required in FIWARE.

5.4.1.2 *Description of GEs*

The mediation GEs are divided in three main parts, as detailed bellow: Data Mediation, Protocol Mediation, and Process Mediation. In any cases, the mediation component should act as a *broker* between service consumers and providers, and it should be as transparent as possible.

Moreover mediation should include some non-functional requirements like built-in monitoring and management capabilities in order to be able to be automatically re-configured and to track mediation steps.

Data Mediation

Various mechanisms can be utilized to perform Data Transformation between different data models employed by the sender and receiver:

- Using a library of built-in transformers in case of well-known formats
- Using templates for more flexible data transformation capabilities
- Using DSL (Domain Specific Language) or code components in order to fulfil more complex data transformation requirements
- Using LILO schema (Lifting and Lowering) and ontology mappings

Data mediation can be used at design time service composition in data flow generation, including data flow mapping and consistency check.

For highly dynamic cases (when services at discovered at runtime, also called *late-binding* of services), data mediation provides the glue between heterogeneous system by:

- Semantic matching, at runtime (also supported at design time), of requested and provided data types. It is based on semantic annotations/meta-data over service descriptions (SAWSDL, USDL), extracted from ontologies (RDF/S, OWL, WSMO, etc).
- Providing semantic matching algorithms that can deal, at runtime, with potential syntactic discrepancies in exchanged data types (from parameters and return values of services operations).
- Increasing, as a result, the agility of systems.
- Integrating with legacy systems (non-intrusive, annotation based).

Service providers are describing and operating web services. Semantic annotations can be put on service descriptions and data-types. Additionally, service providers can issue pure semantic service descriptions using schemas such as OWL-S, WSMO and its light counterparts: WSMO Lite, MicroWSMO. Service consumers are describing their needs in a similar fashion.

Protocol Mediation

- Support for hybrid service compositions that invokes either WSDL based or REST based external services or other protocols based services (for example binary protocols like Hessian or vertical industry standard like FiX and HL7)
- Support, at runtime, for mediation of heterogeneous communication protocols used by service providers and consumers (each integrated system may use its own communication channels/means).
- Provides a combined JBI/ESB environment on which all data will be exchanged and heterogeneous protocols “translated” to the internal protocol of the bus.
- Provides service virtualization and routing capabilities in order to transparently add protocol mediation strategy to a specific service.
- Provides a modular OSGi Environment that enables an easy composition of built-in and custom developed capabilities
- Support for communication protocol extensions such as extending a protocol that allows intra-web browser communication to inter-browser communication.

Process Mediation

- Design time service composition task resolution. Composition tasks are resolved at design time with best matching sub process (service composition) according to the task description (for instance, based on its light semantic description), following a modelling by reused paradigm.
- Deployment time executable service composition generation from a design time service composition model. Support for translation into executable languages, such as BPEL and BPMN 2.0.

- Semantic matching of service capabilities. Capabilities are high-level business features offered by a Web service that are defined as a valid sequence of calls (e.g. a BPEL business process) to several distinct operations of an individual service interface.
- Provides routing capabilities through Enterprise Integration patterns implementation (i.e. message splitting, aggregation and iteration) in order to mediate different processes
- Provides an execution runtime where composite application can be deployed in order to provide more complex process mediation capabilities
- Provides an execution environment where BPMN 2.0 based process can be executed

5.4.1.3 *Critical product attributes*

- Seamlessly providing and requesting services without having to worry about mediation problems.
- Design time and runtime mediation support.
- Process adaptation at design and runtime including dynamic binding, data flow generation and semantic consistency checking.
- Template based generation of processes by reuse.
- Adaptive process deployment, supporting different execution engines.

Existing products

The SETHA2 framework from THALES is a software framework that deals with dynamicity and heterogeneity concerns in the SOA context and is composed of several components/tools that can be deployed independently, depending on the targeted needs of FI-WARE. A major part of SETHA2 is about providing libraries/facilities dedicated to data mediation.

ICT-SERVICE from TI (customization of the WSO2 SOA suite) includes some Data Transformation capability provided by libraries based on the open source project Apache Camel. Currently it does not provide semantic annotation management. It also provides provides a modular execution environment where a composite application runtime and a BPMN 2.0 runtime can be plugged in; moreover it povides some protocol mediation capabilities and a modular OSGI environment that enables easy composition of built-in and custom developed capabilities.

SOA4All Design Time Composer provides some design time semi-assisted modelling features for data mediation and hybrid REST / WSDL based service compositions

PetalsESB is an extensible ESB that supports JBI. It has been successfully deployed in the SemEUsE ANR project and is currently being deployed in the CHOReOS European/FP7 project.

Ericsson Composition Execution Agents: SIP, WS, REST, WARP, RMI, JBI fulfil the automatic translation from from/to such services to the core composition engine input/output. Moreover Ericsson provides the WebCommunication mediator (WARP) to compose browser gadgets also across different device/browser boundaries.

5.5 Generic Enablers for Multi-channel and Multi-device Access

5.5.1 Multi-channel/Multi-device Access System

5.5.1.1 *Target usage*

Nowadays, the huge spread of mobile devices – smart phones, tablets, connected devices of all sorts- and the broad spectrum of social networking platforms have prompted the need for delivering FI applications and services by means of multiple channels (such as social and content platforms, application marketplaces, and so on), while ensuring the best user experience at any time allowing their access from any kind of device (multi-device), adapting usability and presentation when necessary. It is also important to manage user's contextual information in order to support service composition adaptation corresponding to user's preferences and profile. The more detailed and relevant the information at hand and the smarter the ability to reach the end-user the greater are the chances to accelerate time to market, close a sale, or improve customer satisfaction.

User roles

- Application Developer will provide channel specific user interfaces, including the corresponding workflow definition.
- Managers will provide both the content of the knowledge base and access mechanisms to get data from it.

5.5.1.2 *GE description*

In order to support the ideas behind this stage, FI applications must be able to give up the control over their user interfaces to take advantage of an external multi-channel and multi-device access enabler. Applications must provide device-independent abstract user interfaces by means of a standardized user interface definition authoring language, in order to have it properly rendered according to the channel and device's properties and features, publicly available in a shared and standardized knowledge base. Moreover, giving up the control over the user interface also implies the adapter to be on charge of the interface workflow execution, which will be able to call back the application backend through service access points, and control the selection of rendered views.

Apart from solving rendering aspects, which is mandatory for enabling both multi-channel and multi-device access, multi-channel adaptation also requires dealing with the diversity of APIs and capabilities provided by the different channels. More specifically, each channel requires its own specific workflow, and thus there must be support for describing the application workflow in a generic enough abstract workflow language that can be concretized on demand to the target channel.

A workflow engine and a number of renderers, at least one per channel, leveraging the device's properties and the delivery context can tackle multi-device adaptation within a channel.

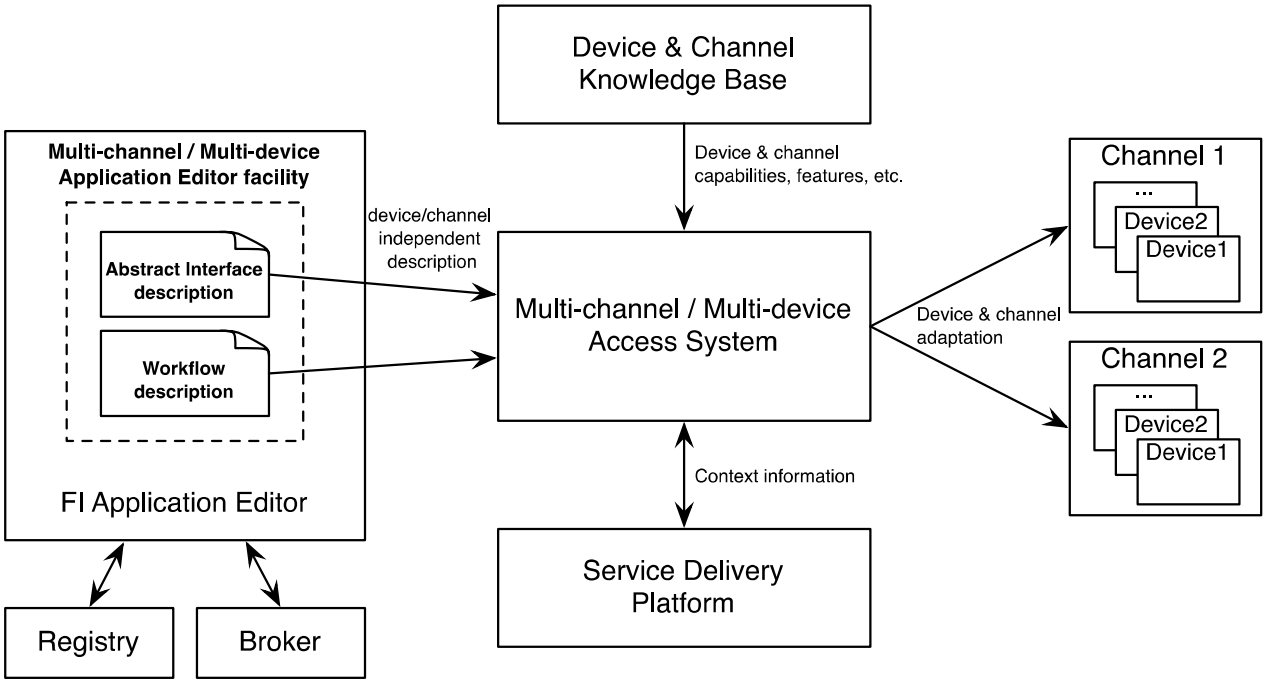


Figure 55 Multi-channel / multi-device Access System

Functionality

- Creating a channel/device-specific user interface from a device/channel-independent definition language.
- Storing and delivering specific data regarding capabilities, features, and constrains of all targeted devices and channels.
- Using data from device description knowledge bases to adapt the user interface.
- Handling the channel specific workflow and perform the backend invocations when necessary, redirecting to the adapted interface.
- Rendering the specific channel user interface according to the Device & Channel Knowledge Base.
- Rendering the specific device user interface according to the Device & Channel Knowledge Base and the targeted channel.

Relations to other components

- Components resulting from chapter "Data/Context Management Services". Specifically user's contextual information managed by the Service Delivery Framework such as user's preferences and profile, user's location, etc.
- Service Delivery Framework

5.5.1.3 *Critical product attributes*

- Access to services and applications from a diversity of devices and distribution channels, including social web networks, content platforms, dedicated web portals, iOS, etc.
- Multi-channel/multi-device access to the FI-WARE platform itself (business framework, repository and registry, and to the application and services themselves from any available delivery context).

- Channels that can be used in a multi-device fashion must get some kind of device id in order to be able to discover its properties into the knowledge base.

Existing products

W3C's initiative DIAL aimed at standardizing an authoring language for device independence.

Products such as MyMobileWeb, HAWHAW, Volantis Mobility Server, Mobile Aware's Mobile Interaction Server, Sevanval's Multichannel Server, Netbiscuits provide a rendering engine based on their own proprietary authoring language.

Cyntelix provides a social media distribution and aggregation platform that addresses multi-channel deployment.

5.6 Question Marks

5.6.1 Security Aspects

This section identifies potential security issues raised by the proposed high-level architecture. The first analysis has shown that the currently available USDL security extension needs conceptual rework. The usefulness of the security enablers to applications and services ecosystems needs further analysis. The architectures, protocols, and description languages used to realize composition and mashup GEs have to be identified and collected to enable in-depth security analysis.

Service registry and repository

The following topics require further analysis:

- Management of identities and authorization for the publication/management of service descriptions in the repository

  Only the owner of the service should be able to define, modify and delete a service description in the registry
- Access control / authentication for the discovery and service search (who can access to a service description)

  Private or corporate services should not be visible for any user

  Users should have a guarantee about the authenticity of the published services, for example an SAP service must be certified and during the search process only certified services should appear to the user if it is required
- Protection against replays

  Replaying discovery requests/response should be forbidden to prevent against Phishing attacks
- Prevent against misusage and coalition attacks for the user feedback system

  If the reputation of a service depends on the feedbacks of non authenticated users, this would lead to positive/negative feedbacks exaggerations. Prevent coalition attack to promote or discredit a service.
- Message Confidentiality between registries, servers, users

Marketplace

The following topics require further analysis:

- Virus and malware scanning for the deployed applications in the "store"

Services that are exposed to the users should be tested against viruses, malwares and adwares to protect the users during the consumption.

- Service signature to authenticate services
- Revocation list maintenance

  If a service had a malicious behavior and was rejected from the marketplace, his clone should not be re-published.
- Authentication for the services

Revenue Settlement and Sharing System

- Confidentiality and integrity during the payment process
- Strong authentication to verify the payment origin and destination
- Privacy protection of the sensitive payment information (Credit cards, traces, logs etc.),
- Accountability to keep trace about the payments
- Secure payment (virtual money ?)

Compostion and Mashup

- Cross domain authentication and access control (federation)

  If services are using different identity providers and roles attributions a federation mechanism should be put in place in order to harmonize the translation between the two domains and preserve the security policy effects.
- Avoid conflicts when composing security policies related to services

  A composite service composed of two services: one encrypting messages and another one passing it in clear will create a conflict.
- Keep the coherence of composed security policies

  Avoid redundancy when enforcing security functionalities: double authentication for example.

Data Mediation

- Data privacy: the mediator should not access to private data

5.6.2 Preliminary Generic Enablers

Store is considered to be an important part of the business framework. However, there are many stores already in commercial use so that an implementation of a store within the business framework will provide no real value. Hence, a store is considered to be an external system that can be integrated into the business framework through the interface to the marketplace. Reference integration with an existing store is envisaged depending on the availability of resources.

Aggregator Repository is probably not a generic enabler on its own. It could be seen as a specific implementation of the Model Repository.

5.6.3 Domain specific extensions for aggregation

In order to augment the orchestration engine towards applicability for IoT-aware services there is a need to also provide extensions to the composition language used in the orchestration engine.

There are idiosyncrasies of IoT services that impose significant differences between current and future business processes that will be IoT-aware. These include among others an inherently distributed execution: The automated and semi-automated execution of a modelled business process is one of the key benefits of process modelling. In contrast to having a central process engine in a WoS process, the execution of the process steps is usually distributed over the devices in an IoT-enabled process. The orchestration of these distributed execution activities must be possible with an IoT-aware process modelling language. Furthermore, IoT deals with distributed data: When business processes are realized in the standard enterprise world, central data storage, e.g. a database server, is normally the only data storage. In the IoT it is possible to distribute the data over several of the resources of a device, potentially even eliminating a central storage. A modelling language must allow arranging this distribution of data. Related to this is the issue of Scalability: In standard business processes there is generally only one central device and resource repository, but in the IoT processes multiple devices and resources (e.g. sensors and actuators of a fridge) can appear. The complexity of the modelled process should be independent from the number of devices, resources, and services. Additionally, the growing number of devices should not have an impact on the performance of the process execution. Therefore, the modelling language must provide concepts to describe the expected performance even for many devices.

There are even more IoT specific aspects to business process modelling such as availability/ mobility, fault tolerance, or the uncertainty of information that is often an issue with information gathered from IoT sensors. Within FI-WARE we will provide IoT notation extensions in a similar way as the USDL extensions.

5.6.4 Relationship to I2ND chapter

Interface to Network and Devices: A number of composition and mashup GEs is expected to consume Telco services. The relationship to Interface to Network and Devices requires in-depth analysi

5.7 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)

- **Aggregator (Role):** Supports domain specialists and third-parties in aggregating services and apps for new and unforeseen opportunities and needs. It does so by providing the dedicated tooling for aggregating services at different levels: UI, service operation, business process or business object levels.
- **Broker (Role):** The business network's central point of service access, being used to expose services from providers that are to be delivered through the Broker's service delivery functionality. The broker is the central instance for enabling monetization.
- **Business Element:** Core element of a business model, such as pricing models, revenue sharing models, promotions, SLAs, etc.
- **Business Framework:** Set of concepts and assets responsible for supporting the implementation of innovative business models in a flexible way.
- **Business Model:** Strategy and approach that defines how a particular service/application is supposed to generate revenue and profit. Therefore, a Business Model can be implemented as a set of business elements which can be combined and customized in a flexible way and in accordance to business and market requirements and other characteristics.
- **Business Process:** Set of related and structured activities producing a specific service or product, thereby achieving one or more business objectives. An operational business process clearly defines the roles and tasks of all involved parties inside an organization to achieve one specific goal.

- **Business Role:** Set of responsibilities and tasks that can be assigned to concrete business role owners, such as a human person or a software component.
- **Channel:** Resources through which services are accessed by end users. Examples for well-known channels are Web sites/portals, web-based brokers (like iTunes, eBay and Amazon), social networks (like Facebook, LinkedIn and MySpace), mobile channels (Android, iOS) and work centres. The mode of access to these channels is governed by technical channels like the Web, mobile devices and voice response, where each of these channels requires its own specific workflow.
- **Channel Maker (Role):** Supports parties in creating outlets (the Channels) through which services are consumed, i.e. Web sites, social networks or mobile platforms. The Channel Maker interacts with the Broker for discovery of services during the process of creating or updating channel specifications as well as for storing channel specifications and channelled service constraints back in the Broker.
- **Composite Service (composition):** Executable composition of business back-end MACs. Common composite services are either orchestrated or choreographed. Orchestrated compositions are defined by a centralized control flow managed by a unique process that orchestrates all the interactions (according to the control flow) between the external services that participate in the composition. Choreographed compositions do not have a centralized process, thus the services participating in the composition autonomously coordinate each other according to some specified coordination rules. Backend compositions are executed in dedicated process execution engines. Target users of tools for creating Composites Services are technical users with algorithmic and process management skills.
- **Consumer (Role):** Actor who searches for and consumes particular business functionality exposed on the Web as a service/application that satisfies her own needs.
- **Desktop Environment:** Multi-channel client platform enabling users to access and use their applications and services.
- **Event-driven Composition:** Components concerned with the composition of business logic which is driven by asynchronous events. This implies run-time selection of MACs and the creation/modification of orchestration workflows based on composition logic defined at design-time and adapted to context and the state of the communication at run-time.
- **Front-end/Back-end Composition:** Front-end compositions define a front-end application as an aggregation of visual mashable application pieces (named as widgets, gadgets, portlets, etc.) and back-end services. Front-end compositions interact with end-users, in the sense that front-end compositions consume data provided by the end-users and provide data to them. Thus the frontend composition (or mashup) will have direct influence on the application look and feel; every component will add a new user interaction feature.
- Back-end compositions define a back-end business service (also known as process) as an aggregation of backend services as defined for service composition term, the end-user being oblivious to the composition process. While back-end components represent atomization of business logic and information processing, front-end components represent atomization of information presentation and user interaction.
- **Gateway (Role):** The Gateway role enables linking between separate systems and services, allowing them to exchange information in a controlled way despite different technologies and authoritative realms. A Gateway provides interoperability solutions for other applications, including data mapping as well as run-time data store-forward and message translation. Gateway services are advertised through the Broker, allowing providers and aggregators to search for candidate gateway services for interface adaptation to particular message standards. The Mediation is the central generic enabler. Other important functionalities are eventing, dispatching, security, connectors and integration adaptors, configuration, and change propagation.
- **Hoster (Role):** Allows the various infrastructure services in cloud environments to be leveraged as part of provisioning an application in a business network. A service can be deployed onto a specific cloud using the Hoster's interface. This enables service providers to re-host services and applications from their on-premise environments to cloud-based, on-demand environments to attract new users at much

lower cost.

- **Marketplace:** Part of the business framework providing means for service providers to publish their service offerings and service consumers to compare and select a specific service implementation. A marketplace can offer services from different stores and thus different service providers. The actual buying of a specific service is handled by the related service store.
- **Mashup:** Executable composition of front-end MACs. There are several kinds of mashups, depending on the technique of composition (spatial rearrangement, wiring, piping, etc.) and the MACs used. They are called application mashups when applications are composed to build new applications and services/data mash-ups if services are composed to generate new services. While composite service is a common term in backend services implementing business processes, the term 'mashup' is widely adopted when referring to Web resources (data, services and applications). Front-end compositions heavily depend on the available device environment (including the chosen presentation channels). Target users of mashup platforms are typically users without technical or programming expertise.
- **Mashable Application Component (MAC):** Functional entity able to be consumed executed or combined. Usually this applies to components that will offer not only their main behaviour but also the necessary functionality to allow further compositions with other components. It is envisioned that MACs will offer access, through applications and/or services, to any available FI-WARE resource or functionality, including gadgets, services, data sources, content, and things. Alternatively, it can be denoted as 'service component' or 'application component'.
- **Mediator:** A mediator can facilitate proper communication and interaction amongst components whenever a composed service or application is utilized. There are three major mediation area: Data Mediation (adapting syntactic and/or semantic data formats), Protocol Mediation (adapting the communication protocol), and Process Mediation (adapting the process implementing the business logic of a composed service).
- **Monetization:** Process or activity to provide a product (in this context: a service) in exchange for money. The Provider publishes certain functionality and makes it available through the Broker. The service access by the Consumer is being accounted according to the underlying business model and the resulting revenue is shared across the involved service providers.
- **Premise (Role):** On-Premise operators provide in-house or on-site solutions, which are used within a company (such as ERP) or are offered to business partners under specific terms and conditions. These systems and services are to be regarded as external and legacy to the FI-Ware platform because they do not conform to the architecture and API specifications of FI-Ware. They will only be accessible to FI-Ware services and applications through the Gateway.
- **Prosumer:** A user role able to produce, share and consume their own products and modify/adapt products made by others.
- **Provider (Role):** Actor who publishes and offers (provides) certain business functionality on the Web through a service/application endpoint. This role also takes care of maintaining this business functionality.
- **Registry and Repository:** Generic enablers that able to store models and configuration information along with all the necessary meta-information to enable searching, social search, recommendation and browsing, so end users as well as services are able to easily find what they need.
- **Revenue Settlement:** Process of transferring the actual charges for specific service consumption from the consumer to the service provider.
- **Revenue Sharing:** Process of splitting the charges of particular service consumption between the parties providing the specific service (composition) according to a specified revenue sharing model.
- **Service Composition:** in SOA domain, a service composition is an added value service created by aggregation of existing third party services according to some predefined work and data flow. Aggregated services provide specialized business functionality on which the service composition functionality has been split down.
- **Service Delivery Framework:** Service Delivery Framework (or Service Delivery Platform (SDP))

refers to a set of components that provide service delivery functionality (such as service creation, session control & protocols) for a type of service. In the context of FI-WARE, it is defined as a set of functional building blocks and tools to (1) manage the lifecycle of software services, (2) creating new services by creating service compositions and mashups, (3) providing means for publishing services through different channels on different platforms, (4) offering marketplaces and stores for monetizing available services and (5) sharing the service revenues between the involved service providers.

- **Service Level Agreement (SLA):** A service level agreement is a legally binding and formally defined service contract between a service provider and a service consumer, specifying the contracted qualitative aspects of a specific service (e.g. performance, security, privacy, availability or redundancy). In other words, SLAs not only specify that the provider will just deliver some service, but that this service will also be delivered on time, at a given price, and with money back if the pledge is broken.
- **Service Orchestration:** in SOA domain, a service orchestration is a particular architectural choice for service composition where a central orchestrated process manages the service composition work and data flow invocations the external third party services in the order determined by the work flow. Service orchestrations are specified by suitable orchestration languages and deployed in execution engines who interpret these specifications.
- **Store:** An external component integrated with the business framework offering a set of services that are published to a selected set of marketplaces. The store thereby holds the service portfolio of a specific service provider. In case a specific service is purchased on a service marketplace, the service store handles the actual buying of a specific service (as a financial business transaction).
- **Unified Service Description Language (USDL):** USDL is a platform-neutral language for describing services, covering a variety of service types, such as purely human services, transactional services, informational services, software components, digital media, platform services and infrastructure services. The core set of language modules offers the specification of functional and technical service properties, legal and financial aspects, service levels, interaction information and corresponding participants. USDL is offering extension points for the derivation of domain-specific service description languages by extending or changing the available language modules.

6 Internet of Things (IoT) Services Enablement

6.1 Overview

Contemporary society is demanding more services based on smart environments all the time. Smart grids, smart metering, home automation, eHealth, logistics, transportation, environmental monitoring are just a few examples of the new wave of services we will be widely using in the following years. These solutions will be driven by the Internet of Things (IoT) where the power of combining ubiquitous networking with embedded systems, RFID, sensors and actuators makes the physical world itself a relevant part of any information system. RFID has played a big role in establishing the concept of an Internet of Things – the term was first introduced by the MIT Auto-ID Center in 1999, the precursor to the current EPCglobal organisation, and at that time stood for the vision of a world where all physical objects are tagged with an RFID transponder with a globally unique ID. Based on this initial work, companies like WalMart in the US and Metro in Europe have built new relationships with their more than 5,000 suppliers to improve supply-chain operations. Technological improvements in RFID and the inclusion of sensing and other technologies have expanded the scope of the Internet of Things and are leading to many new opportunities in the way how communicating things can support new generation lifestyles. In the short-term, it is estimated that the number of M2M devices outnumber end users by at least one order of magnitude [Juniper 11] and some industrial projections indicate that the world market for technologies, products, applications and enabled smart services related to IoT will increase significantly to more than €290 billion by 2014 [Harbour 10]. The relevance of the IoT is not only recognized in the EU, as shown by the report [IoT Brussels 09], Japan, China and Korea already include IoT initiatives in their national R&D programs, and the U.S. National Intelligence Council has included the IoT among the six technologies with potential impacts on U.S. interests out to 2025. The important role of the Internet of Things is also highlighted by the fact that many international standard-developing organizations have established dedicated standardization initiatives on the topic: ETSI TC M2M, ITU-T USN, ISO/IEC WGSN, IETF, W3C, 3GPP.

- **Thing.** Physical object, living organism, person or concept interesting from the perspective of an application.

To address the important challenges and opportunities the IoT represents, FI-WARE includes the Internet of Things Service Enablement as one of the chapters to be addressed within the Reference Architecture of the FI-WARE Platform. The IoT Service Enablement chapter comprises those Generic Enablers in FI-WARE enabling a large number of distributed and heterogeneous things and associated IoT resources to become available, searchable, accessible and usable by Future Internet Applications and Services.

Several key technologies must still be developed enabling the Internet of Things vision to become a reality. Current services and technologies for accessing real-world information are typically closed leading to vertically integrated solutions for niche applications. This approach leads to inefficient and expensive service infrastructures that lack interoperability and prevent the true IoT paradigm from being successfully developed. Obviously each vertical domain of business applications can have various types of peculiarities, nevertheless development of cross-domain horizontal solutions, including common functionalities, would lead to more efficient and cost effective solutions. Furthermore, in this way, an infrastructure deployed for use by a vertical solution, can be shared to provide completely different services, opening new business opportunities based on innovative business models. Following this direction the IoT Service Enablement in FI-WARE will provide a set of IoT-oriented Generic Enablers further classified into sub-chapters dealing with: IoT communications, heterogeneous resource management, data handling and process automation. This substrate of IoT-oriented Generic Enablers will provide important efficiency gains in many industries and usage areas, particularly when combined with other FI-WARE Generic Enablers.

The term **IoT resource** refers to the computational elements (i.e. software running on devices), enabling to gather information about things and act upon them. The number of devices is expected to outnumber the human population by orders of magnitude [Juniper 11], thus the development of efficient means for communication, management and interaction with them will be a necessity, coping with the need to support the co-existence of a variety of existing and emerging communication technologies. Actually, the IoT application environment is extremely heterogeneous, leading to different interaction patterns (push, pull, converge-cast, multicast, periodic, event-based, etc.) and different interaction constraints compared to the current Internet (in terms of reliability, timing, capabilities of devices, etc.) This heterogeneous application environment will thus also entail a pronounced heterogeneity of the underlying communication layers. The IoT Communication function in FI-WARE will allow unified communications regardless of the different network standards and will enable data transfer services agnostic to the underlying connection protocol.

- **Device.** Hardware entity, component or system that either measures properties of a thing/group of things or influences the properties of a thing/group of things or both measures/influences. Sensors and actuators are devices.
- **IoT Gateway.** A device hosting a number of features of one or several Generic Enablers of the IoT Service Enablement. It is usually located at proximity of the devices to be connected.
- **IoT Resource.** Computational elements (software) that provide the technical means to perform sensing and/or actuation on the device. The resource is usually hosted on the device.

Together with communication protocols abstraction it will also address communication service capabilities, discontinuous connectivity, mobility, session management, traffic flow management as well as access security policy control.

Scenario 1. *Dynamic association, roaming*

Winter 2012 – 7:25 AM, starting the car, David's car device asks for activation regarding traffic and pollution applications. David chooses traffic application and the car device sends a signal to the closer gateway that the vehicle is active and in the traffic. The gateway, based on the planned route home-office defined in the car device, diffuses to the gateways network that a new actuator is alive and the expected time instant when it will roam from the first area (gateway 1) to the second area (gateway 2).

Being able to identify, discover and manage IoT resources is not an add-on but a basic need. However, the devices linked to these resources that have to be connected are per se very heterogeneous in terms of used technology, protocols, capabilities and interaction patterns. Currently, each technology (i.e. ZigBee, 6LoWPAN, etc.) provides some of these functions in different ways. FI-WARE will integrate heterogeneous identification, naming and addressing technologies using unequivocal identifiers. Scalable discovery and look-up will make IoT resources available to all type of applications considering important real-world aspects like location, time, availability, capabilities, quality, etc. It will also provide the necessary functionality to monitor dynamic links between IoT resources and things. Finally, FI-WARE will enable IoT resources to become citizens of the Internet by providing scalable global schemes for deployment, operation, maintenance and fully remote management of these resources, including abstract models for the resources, common resource management interfaces, status monitoring, and fault handling.

Scenario 2. *Sensor measurement sharing*

During the journey to David's office, the car device receives a short message from gateway 7 to activate weather data collection to draw snow storm progress on the city map. As David is driving, the car device launched a request to the profile database to validate David choices. Based on the latest information, from yesterday 10:00 AM, David has no objection to communicate weather information. Immediately, the car

device begins to send temperature, windscreen activity, humidity level…

The IoT realization will imply managing a huge number of highly distributed IoT resources running on devices, which are continuously generating a large amount of data. Efficient data processing mechanisms executing near or adjacent to the resources and generating a smaller set of elaborated data may be necessary, preventing the flooding of the backend system and the storage of large amounts of raw data. In addition, data filtering and routing capabilities have to be placed along the communication path between resources and the backend systems, dynamically adapting the traffic of data to the real demand of application/services at any given moment. The development of new application and services will then rely on the capability to extract information from processed/filtered data applying new generation mining and analysis techniques. FI-WARE will include IoT-oriented Generic Enablers, which would implement the same or a subset of the interfaces defined for FI-WARE Data/Context Management Generic Enablers (Publish/Subscribe GE, Complex Event Processing GE) but would be better suited to run on distributed devices or device gateways and exploit P2P mechanisms. Furthermore, due to the specificity of data handling in de-centralized IoT environments, the IoT Generic Enablers will address data models for dedicated protocols, data models integration and manage relationships between micro databases hosted by IoT resources and other database services. The combination of Generic Enablers running both in distributed locations and centralized backend systems will allow a scalable, elastic and high performing management of the entire Internet of Things.

Scenario 3. *Data aggregation*

Tomorrow: David wants to go sooner to his office because the home clock screen advertises that weather forecast was too optimistic today and that bad weather will come during the night. The picture, based on 357 climate sensors, shows clearly that the snow storm will arrive around 7:30 AM. When he begins to cook his dinner, his home gateway informs the city eco-management application that a new consumption cycle is planned in this district. All cities till around 100 km from his home are sending information. The 357 sensors indicate clearly that air pressure is falling quickly and some mobile sensors in this area are providing real-time temperature indications. Night would be very cold and a snow storm is now forecast for tomorrow evening.

The full potential of the IoT can only be unleashed when IoT resources and data are composed, aggregated and integrated into processes fulfilling specific business needs. For scalability, this has to happen in an automated and distributed manner. To do so, the IoT Sevice Enablement chapter in FI-WARE will provide several Generic Enablers targeted to support a de-centralized automation of micro-processes, managed over different elements at the edge of the network. These Generic Enablers will be capable to deal with lower granularity, locality of execution, quality of information aspects etc. They will extend other FI-WARE Generic Enablers dealing with orchestration and composition at a higher level, thus providing the secure, configurable and scalable end-to-end process automation required in future IoT Applications.

Summarizing the above, **FI-WARE will build the relevant Generic Enablers for Internet of Things Service Enablement, in order for things to become citizens of the Internet – available, searchable, accessible, and usable – and for FI services to create value from real-world interaction enabled by the ubiquity of heterogeneous and resource-constrained devices.**

FI-WARE has been conceived as an end-to-end open platform intended to be used in a broad range of IoT application scenarios and by different kinds of users which are the following:

- FI-WARE IoT instance providers: they provide value added and/or managed services towards IoT customers from different usage areas. For example they could be system integrators or service companies offering turn-key FI-WARE instance installation projects to a government or an enterprise or operating FI-WARE instance as a managed service.
- IoT application developers: they develop end customer applications utilizing APIs/SDKs. For example, home energy management applications for consumers or for utility companies.

- IoT end customers: they utilize IoT services as part of their business processes. Alternatively, they can be individual consumers with no programming skills that need to capture and make sense of data and events
  - from the things through the associated devices by means of IoT resources as input to Internet applications and services
  - from the applications and services to drive things through the associated devices (actuators) by means of IoT resources

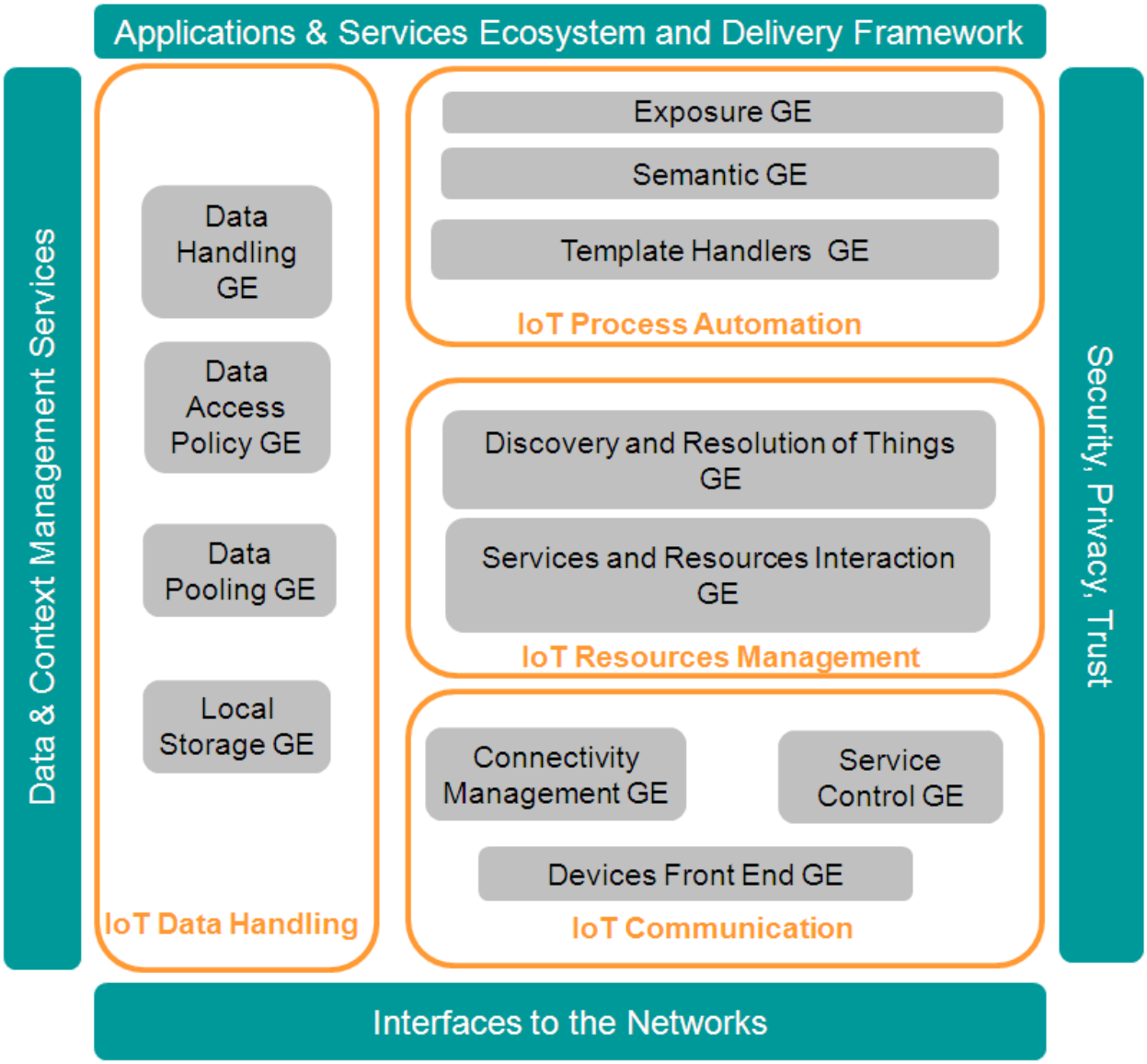


Figure 56 GEs of the IoT Service Enablement platform and its connections to other technical chapters

6.2 Generic Enablers

The deployment of the architecture of the IoT Service Enablement chapter is typically distributed across a large number of devices, several IoT Gateways and the IoT Backend. The Generic Enablers described in this chapter, shown in Figure 1, implement functionalities distributed across IoT resources hosted by devices, IoT Gateways and in the IoT backend.

The IoT Gateway and IoT Backend will interface with each other through an open standardized communication interface. In addition, the IoT gateway will provide a protocol adaptation and control service that will contribute to supporting heterogeneity in the physical world. The protocol adaptation service will handle communication with different devices, while the control service will contain intelligent control logic that will deal with differences between the various implementations of the Gateway and access technologies, as shown in Figure 3.

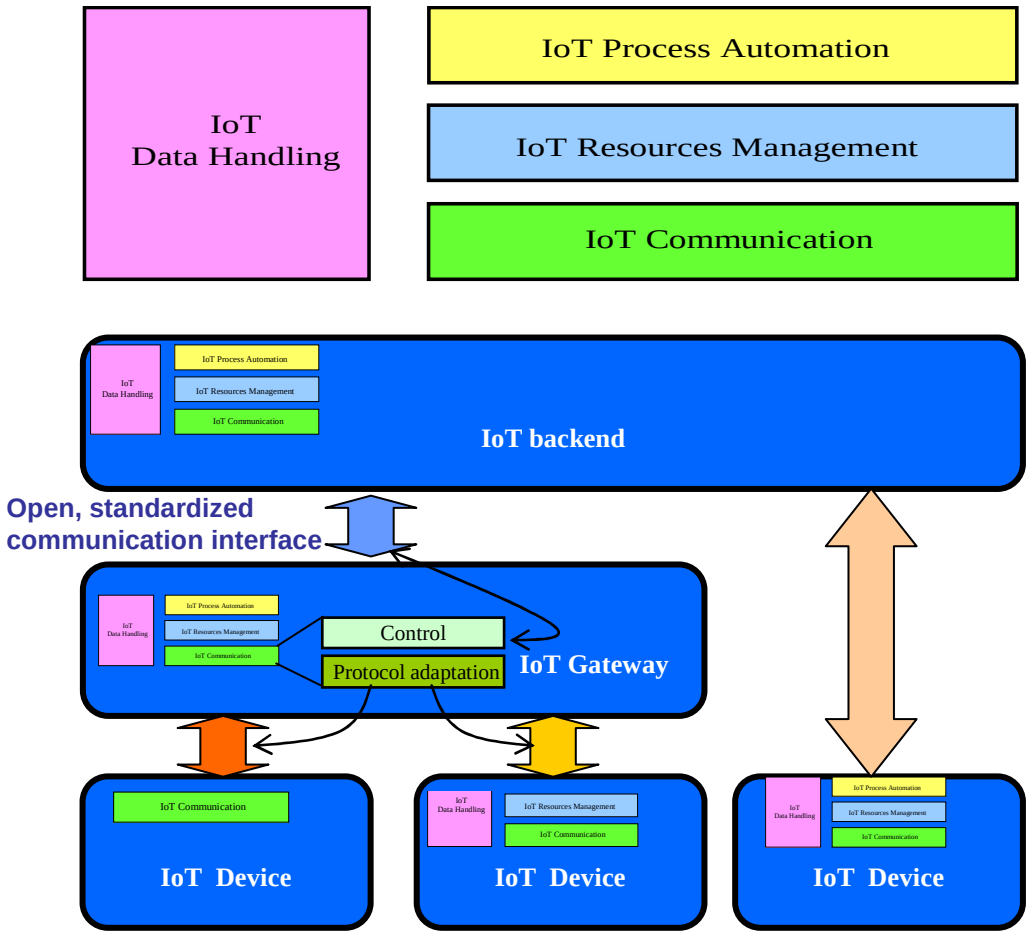


Figure 57 IoT SE "stack" (above), instances of the stack in the backend, IoT gateway and various devices and the control/protocol adaptation service in the IoT gateway (below)

Management functionalities will be provided for the devices and IoT domain-specific support will be provided for the applications. In order to do so, the IoT Service Enablement adopts and integrates the following four technical sub-chapters, each integrating a set of related **IoT Generic Enablers**:

- IoT Communications
- IoT Resources Management
- IoT Data Handling
- IoT Process Automation

The following section elaborates on each of these subchapters, describing their target usage, the set of IoT Generic Enablers that compose them and the list of Critical product attributes were added-value is provided.

6.2.1 IoT Communications

6.2.1.1 *Target usage*

IoT communications will provide common and generic access to every kind of things regardless of any technological constraint on communications, typically integrating several protocols and manage discontinuity of connectivity for nomadic devices. It will allow Application Providers to gain homogeneous access to dedicated things and devices and to be able to manage Quality of Service in Communication with the devices.

6.2.1.2 *Description of GEs*

A communication abstraction layer will be specified in order to integrate different underlying protocols used by various devices and gateways,. This will be based on a uniform protocol view and will be generalised to cover common solutions and technologies in the underlying communication layers.

As a result of the investigation of the state-of-the-art of communication technologies, only a few approaches were found and most of them have not strived for the generalisation deemed necessary for IoT. These are networked RFID systems, wireless sensor networks and machine-to-machine (M2M) communications.

Networked RFID systems use two different communication stacks. The reader-to-tag communication is governed by RFID protocols, while the communication stack between reader and the business logic varies according to the reader's specification. In the last few years readers have evolved providing better and more complex interfaces to edge and middleware software. Currently, the state-of-the-art readers are also capable of processing raw data, providing information via Secure Socket Layer or Web services. Standalone reader abstraction protocols such as the EPCglobal Reader Protocol (abandoned by EPCGlobal, even though ratified) or WinRFID [Prabhu 06] exist while many middleware solutions leverage on reader device-specific device abstraction drivers. The approaches regarding RFID are determined by relevance of the generalised service and communication interfaces.

The distributed sensor networks and in particular Wireless Sensor Network (WSN) platforms are increasingly used as enabling technologies to create underlying layers of IoT systems and the Real World Internet. These platforms usually implement at the communication layer protocols such as 6LoWPan or at the physical layer IEEE 802.15 based protocols. The current sensor nodes are mainly equipped with radio interface to enable communication between them. This interface is a low power, low range radio system covering physical layer and media access layer. Communication protocols such as 6LoWPAN enable sensor nodes to be directly accessible through the Internet, but some platforms do not support this protocol. The heterogeneous nature of WSNs requires a flexible and interoperable solution to enable seamless communication and interaction between high-level application and services and the underlying network, the platforms existing today are not able to provide these kinds of solutions.

An area of investigation that is closer to IoT needs is machine-to-machine (M2M) communication. M2M interfaces have been originally envisaged to describe the communication between electronic devices used for special purposes, such as sensing, metering and control. Recognizing the importance of M2M technologies, the ETSI in 2009 launched the Machine-to-Machine (M2M) Technical Committee [ETSI-M2M]. The goals of the ETSI M2M committee include developing and maintaining of an end-to-end architecture for M2M, identifying the gaps in current standardization and filling these gaps, targeting sensor-network integration, naming, addressing, location, QoS, security, charging, management, application interfaces and hardware interfaces.

The following key M2M technology trends could be highlighted: dealing with an increased complexity of internetworking M2M networks, devices and data with enterprise applications, networks, devices and data; migration from M2M proprietary bus architectures to Internet-based open standards; evolving from centralized, decentralized and distributed computing architectures towards dynamic and hybrid

architectures based on intelligence at the edge of the network. This is a shift in process control enabling, in some cases, autonomous or semi-autonomous control actions to emerge and/or be determined at the edge of the network independent of human or "server" authorisation. Distributed or even autonomous process control however poses challenges regarding management of software, firmware or execution rules as well as overall system reliability.

IoT communication Generic Enablers will utilize the above-mentioned technologies that support communications between distributed things and devices (e.g. gateway and middleware solutions) and will develop standard interfaces and interoperable solutions to coordinate the communication and interaction between things and high-level applications and services. They will also take into consideration the resource-constrained nature of devices and will adopt energy-aware and efficient mechanisms to provide communication solutions between devices, services and applications.

More specifically, the IoT Communication Generic Enablers will provide the necessary communication agents that can be integrated into the IoT Devices and the IoT Gateways, enabling communication with IoT resources from the IoT backend. -

Figure 58 shows the main Generic Enablers implementing IoT Communication functions. These Generic Enablers will provide altogether at least the following functions:

- Communication protocols abstraction: a mechanism to enable unified communications between the IoT resources and the IoT backend
- Communication service capabilities: identification of and communication to IoT resources (identification of an IoT resource includes the mapping of physical network addresses to the IoT resource identifier)
- Managed connectivity: definition of the interfaces towards the networks for the management of the connectivity
- Discontinuous connectivity: management of the IoT resources that are not always-on.
- Mobility: support of the IoT mobility for the IoT resources that may physically move or may change the access network (e.g.: fixed or mobile, IP or SMS,).
- Session management: management of the communication sessions to support mechanisms to handle reliability associated to network connections
- Traffic flow management: development of the mechanisms to deal with abnormal and occasional traffic conditions (e.g. overload traffic conditions)

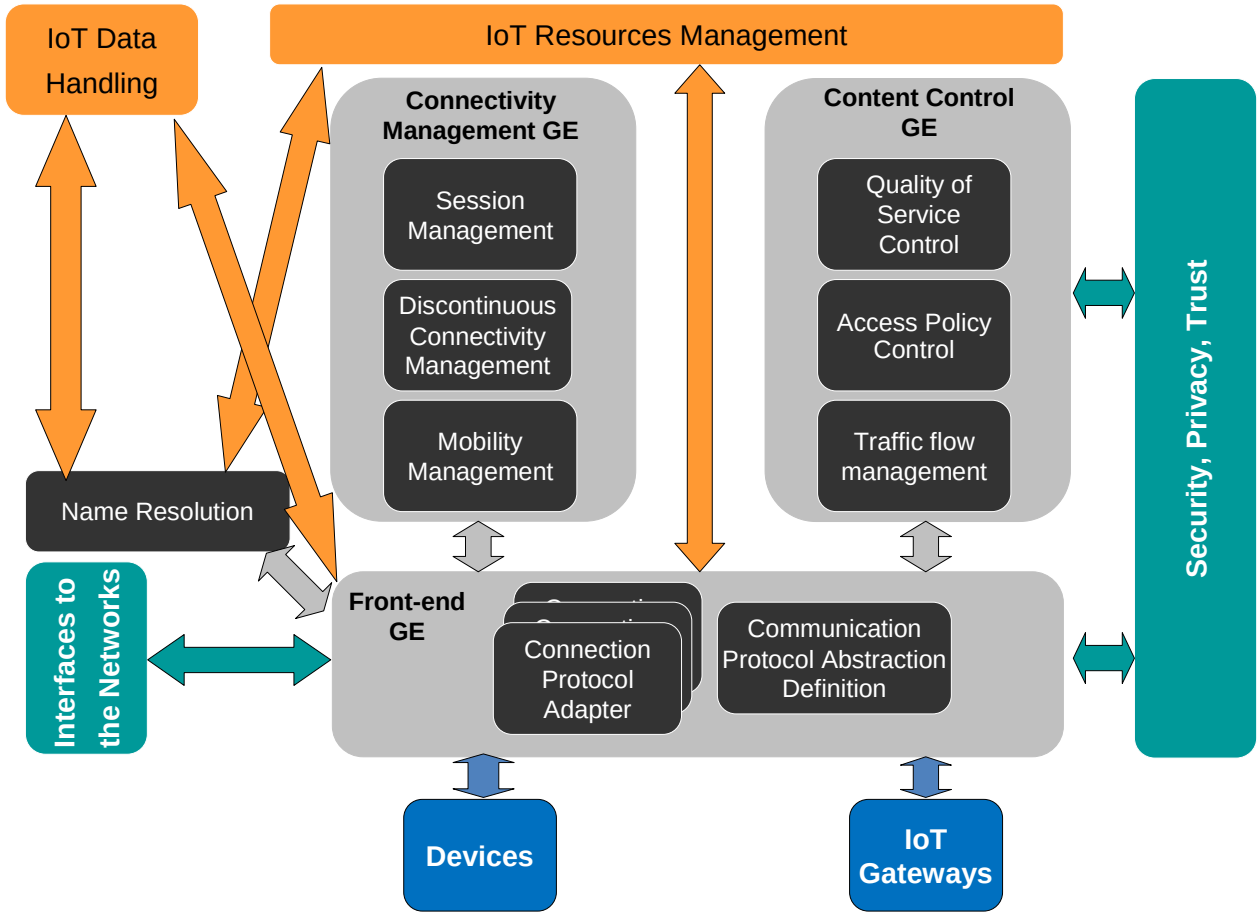


Figure 58 Architecture of IoT Communications and its 4 GEs

The architecture supporting the IoT Communications functions relies on the existence of four Generic Enablers: Front-end, Naming, Connectivity Management, and Content Control.

The Front-end GE deals with the incoming/outgoing traffic from/to Devices and IoT Gateways. It comprises a number of Connection Protocol Adapters and a component dealing with the Communication Protocol Abstraction Definition. Each of the Connection Protocol Adapters is capable of handling one of the protocols that are accepted in FI-WARE, translating it into a unique internal language, which normalizes the different communication protocols within the platform. The definition of this "internal language" is contained in the Communication Protocol Abstraction Definition that provides a number of "templates" that can be used to translate the protocol (e.g. this translation can be achieved by creating an object, i.e. a "token", specific for each incoming message type, containing all the data contained in the incoming message, that can be "injected" into the platform; likewise an outgoing token can be translated into a protocol message that will be sent by the platform to the specific Device or IoT Gateway).

The Front-end GE will rely on Security, Privacy & Trust Generic Enablers in order to perform the necessary management of security aspects. As an example, the Front-end GE relies on Security Generic Enablers to decrypt and encrypt the traffic incoming into and outgoing from Devices and IoT Gateways (e.g. managing of the asymmetric cryptography based on public and private keys). It also relies on Security Generic Enablers coping with management of access rights as it will be described later. Additionally, it relies on the IoT Data handling for storing and retrieving the templates from the Communication Protocol Abstraction Definition.

Moreover the Front-end GE exploits also the functions of the Name Resolution GE. This is a GE that can be shared between IoT Communication and IoT Resources Management. For the IoT Communication it

translates the physical addresses received by the network into the addresses used internally within the platform and gives support to IoT Resources Management too for other address related functions.

The Connectivity Management GE deals with lower level layers of the communication from/to Devices, IoT Gateways and the Content Control block with higher level layers. It consists of components dealing with Session Management, Mobility Management and Discontinuous Connectivity Management. The Session Management controls the communication session between the IoT Services Enablement platform, the Devices and IoT Gateways. The Mobility Management deals with the Devices and IoT Gateways mobility, both the connection through different Access Networks and the physical mobility of the "things". The Discontinuous Connectivity Management allows communicating between IoT Gateways and Devices that are not always on. It manages especially the traffic towards them that may need to be cached waiting for the Device or IoT Gateway to wake up.

The Content Control GE is composed by components dealing with Traffic Flow Management, Access Policy Control and Quality of Service Control. The Traffic Flow Management monitors the overload conditions of the IoT Services Enablement platform. In case the platform is under stress conditions it can command the Front-end GE to redirect or reject incoming messages without any further treatment but log the received messages (this component can affect only incoming traffic). The Access Policy Control and the Quality of Service Control components deal with the permissions to access the IoT Resources (the first one basing the control on the identities of the requestor, of the requested resource and of the requested operation, and the second one on a contractual agreement (Service Level Agreement – SLA) on a quality of service (QoS) applicable to the requestor or to the requested resource). Both controls can be performed either at IoT Communication level or at IoT Resource Management level. If applied within the IoT Communication they can prevent the IoT Resource Management to receive traffic that will be rejected, otherwise all the traffic can be passed to the IoT Resource Management applying these kinds of controls. In any case the Security, Privacy & Trust will provide relevant mechanisms for access rights since they are strictly related with privacy issues.

6.2.1.3 *Critical product attributes*

- Common connectivity management and service interfaces to access, control and manage communications with and about physical things;
- Interworking of things protocols including mobility management, disconnected operations, virtualization of resources, overlay management and security
- Opened and easy to use APIs and tools for connecting heterogeneity of physical things

6.2.2 IoT Resources Management

6.2.2.1 *Target usage*

IoT Resources Management will propose unified service and operational support management functions enabling the different IoT applications and end users to discover, utilise and activate small or large groups of IoT resources and manage their properties. In doing so, the IoT Service Enablement will focus on global identification and information model schemes for IoT resources, providing a resolution infrastructure to link them with relevant things and developing a common remote management tool for configuring, operating and maintaining the IoT resources on a large scale and with minimum human intervention.

A particular challenge represents the handling of heterogeneous identification schemes that will unavoidably exist in an IoT infrastructure and the heterogeneous addressing schemes that will be used to access the devices. IoT resources can be identified by different identification schemes, e.g. RFID based identities such as EPC, UID etc. From the perspective of addressing schemes MAC or IP addresses or other device identifiers such as HIP may be deployed. Similar heterogeneity can be expected from the assignment of identifiers to the real world entities.

In order to find information and services associated to objects in the Internet of Things, it is necessary to have a resolution infrastructure available, which enables the unequivocal and universal identification of both things and IoT resources (and implicitly the device hosting them), and their mapping into network resources. The Name Resolution GE mentioned in the above section is related to this functionality, but positioned on a slightly different level. In IoT Resources Management the resolution is more about how to find an IoT resource of particular properties or an IoT resource associated to a particular thing. In IoT Communications it is about the way to find the physical device (address, port, protocol etc) that hosts this particular IoT resource.

Maybe the most prominent example of such infrastructures is given by the so-called "Object Name Service" (ONS) in the EPCglobal Network. The ONS shares the same hierarchical design as the Internet Domain Name System (DNS), with queries being delegated from a global root server down to local instances of individual organizations. Because the ONS is allowed only to point to the manufacturer's EPCIS repository, the EPC Network includes EPC Discovery Services. These services allow applications to find third parties' EPCIS repositories across an object's individual supply chain, which can then provide detailed event information to others. It is likely that multiple discovery services will coexist in the future. However the EPC Network is just one example for a resolution infrastructure and mainly used in the retail and consumer products industries, while other resolution infrastructures have been proposed and are also being used, as for example the ucode and associated services originated in Japan.

Beyond identification, resource modelling is a key issue for lookup and discovery. Many state of the art proposals for modelling sensors and sensor networks have been evolved from particular application sectors, specific devices or wireless sensor network technologies.

The look-up and discovery of the IoT resources can be performed by searching in the resource descriptions. Ideas from Web search and concepts of semantic search can contribute to the development of efficient look-up and discovery mechanisms. UDDI [UDDI 04] and other methods for discovery of Web services can be extended to provide IoT resources look-up and discovery.

IoT resources management can consider current state-of-the-art in autonomic computing architectures fundamentally as a control loop, self-management research and self-awareness properties, in particular self-optimisation; -organisation; -configuration; adaptation/contextualisation; -healing; -protection.

IoT Service Enablement will rely on the current naming and resolution infrastructures defined in different projects (e.g. Bridge, SENSEI and IoT-A) by introducing scalable customizable information and resource modelling and discovery mechanisms with relations to the Usage Area projects scenarios. It will also develop mechanisms to provide (automated) associations between IoT resources and the things.

6.2.2.2 *Description of GEs*

Figure 59 shows the main Generic Enablers implementing IoT Resource Management functions. These Generic Enablers will provide altogether at least the following functions:

- IoT resources identification: through an unequivocal identifier in order to address exiting known resources using a key or identifier (also known as look-up) for representing a common ground for resource management.
- IoT resolution as a way where a device ID or a service ID is mapped to an address or locator that will allow the communication with the device or service.
- IoT discovery for finding unknown resources/services based on a rough specification of the desired result by IoT applications. Based on unique identifiers, UIDs, an IoT resolution infrastructure will be

able to resolve names and identities to addresses and locators used by communication services. IoT requires the development of resource discovery services to make things available and discoverable to applications.

- IoT resource information model: unified information models to describe IoT resources will be proposed to enhance availability and accessibility for services and end-users. Many IoT applications will require not only a static description of resource capabilities but also dynamic descriptions based on their interaction with other resources and things, thus enabling a true federation of things.
- IoT resolution infrastructure: intended to provide the necessary functionality for discovery, look-up and monitoring dynamic links of IoT resources with things, obtaining the hierarchical dependencies and context between the different elements, and next, applying semantic technologies be able to provide the associated values to one element of the network, directly through it or indirectly thought other container elements of its network.
- IoT linking mechanism: sensors, actuators and personal mobile devices are used to interact with things. An efficient linking mechanism should consider availability and mobility aspects for both for the things and devices. In some cases the ability of IoT resources to interact with things relies on sensors and actuators physically attached to them. In other cases, particular real-world contexts that in particular translate to properties of the virtual entities can be derived from the collaborative interaction between several IoT resources.
- IoT resources management: in the very dynamic scenario of IoT, IoT resources can evolve with varying degrees of autonomy integrating different technologies over different infrastructure providers. To guarantee an efficient and effective deployment of IoT, global schemes for operation and maintenance management should be developed.
    - Fully remote management enabling the initial configuration, activation, software update, de-activation, and re-activation of IoT resources will facilitate the IoT deployment.
    - Using abstract models as a framework for understanding significant relationships among the IoT of some environment. It enables the development of a specific reference model or a concrete architecture using consistent standards or specifications for supporting the environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A common configuration and settings could be easily applied to different groups of resources, including enterprise specific groups, and things type specific groups.
    - Common resource management, relying on open tools and standardized management interfaces, will ease the integration of connectivity modules as well as software management: firmware updates, operating systems, and application logic.

Remote status monitoring will track operational and connection status of resources into the IoT. Fault handling for IoT resources should consider integrated alarm events management and on-demand remote diagnostics to allow specific fault recovery from resources failures.

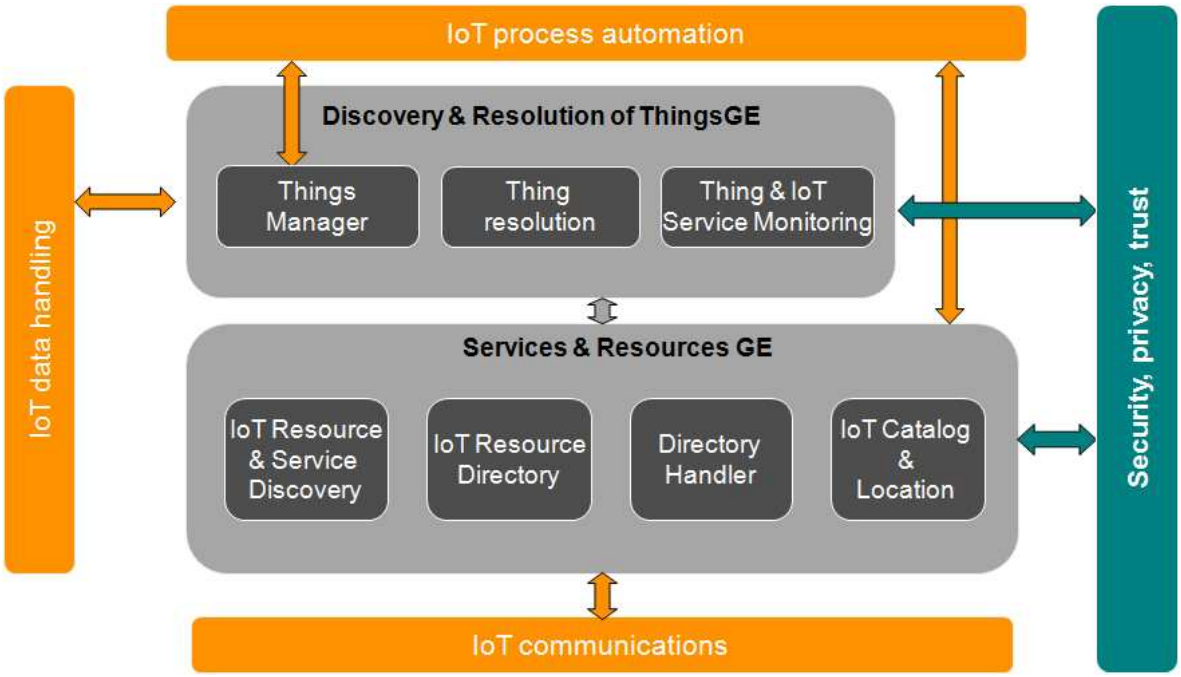


Figure 59 Architecture of IoT Resource Management and its 2 GEs

The architecture supporting IoT Resources Management functions relies on the existence of two Generic Enablers: the Discovery and Resolution of Things GE and the Services and Resources Interaction GE.

The Discovery & Resolution of Things GE will provide functionalities for retrieving lists of services that expose resources related to particular things. This GE will be based on the following components:

- A Thing Resolution component that will provide the functionality to discover the thing based on a general description discovery, if the thing is not clearly identified by the user.
- A Thing and IoT Service Monitoring component that will dynamically maintain associations between Things, IoT resources, and exposed supervision and management services. This component is supported by Service and Resource Interaction GE to retrieve the thing to which resources are associated.
- A Things Manager component that will manage things-related information, including the insertion, update, and deletion of things descriptions and the static association among them and resources. It supports the Service and Resource Interaction GE to manage how IoT resources can be observed and exposed by a given device and, for instance that devices running in sleep mode may no longer hosts all the resources.

The functionality provided by the Services and Resources Interaction GE is offered in two ways, synchronous and asynchronous, by means of the following functional components:

- The IoT Resource Directory component is responsible for keeping all the information of the different devices registered in the IoT service enablement, like physical address, status, location, etc
- The IoT Directory Handler component acts as interfaces between the system and IoT Resource information providers, throught IoT communication. It maintains and provides information regarding an identified IoT resource, allowing to update the description of a resource, to retrieve its description or to provide the address of a identified service. Furthermore it is in charge of the automatic resource registry in the IoT Services & Resource Directory. Through the interface with IoT Communication, this component provides access to the device communication and connectivity status and resources can be retrieved from devices.
- The IoT Catalogue and Location component provides mechanisms in a distributed environment to discover which of the different instances of the entities can support the request a user might be

interested in. For example, in an architecture where several Resource Description components exist, a client might be interested in a particular IoT Resource. The client should interrogate the CLB to know which particular existing Resource Description components contain the information requested.

- The IoT Resource & Service Discovery will interact with IoT Process Automation to provide access to services associated to things, allowing web services to find these resources. This collaboration will manage mostly dynamic discovering new associations between IoT resources and associated services, supporting discovery qualifiers such as location, proximity and other context information. The Resource & Service Discovery will publish to Data/Context Management GE integrated context information about things and their associations (dynamic and static) with other things and resources based on information provided by IoT Data Handling. An interface between Resource & Service Discovery and Security, Privacy and Trust Generic Enablers will support checking access permission before publishing the context information and providing access to resources. A trusted authority will be accessed to verify that only resources provided by trustworthy entities are considered.

6.2.2.3 *Critical product attributes*

- Uniform access to the things in the virtual world and transparent mapping to a real world access
- Sophisticated "Resolution infrastructure" to discover, identify and access information about things through heterogeneous identification systems
- Range of sophisticated look-up mechanism that enable finding the correct things in the millions of available with a large range of selection criteria like object types and type hierarchies, geographic areas of interest, properties of things, relationships between things, semantic tags

6.2.3 IoT Data handling

6.2.3.1 *Target usage*

IoT Data Handling will be essential for Application and Data Service Providers to collect large amount of IoT-related data, produced by a huge number of IoT resources almost in real-time.

IoT Data Handling GEs will be supported by secured and privacy policies in collecting and forwarding data to Application/Services.

The IoT Data Handling sub-chapter will comprise GEs able to handle efficient data representation formats (including semantic data) and execute near or adjacent to the IoT resources. They will be able to execute data processing functions aiming to generate a smaller set of elaborated data locally, preventing the flooding of the backend system and the storage by large amounts of raw data. These GEs can be placed along the communication path between resources and the backend systems, dynamically adapting the traffic of data to the real demand of application/services at any given moment. IoT Data Handling GEs will therefore include IoT-oriented Generic Enablers offering the same or a subset of the interfaces provided by the Publish/Subscribe Broker and Complex Event Processing GEs defined in the Data/Context Management chapter of FI-WARE. However, they may be implemented as complementary products tailored to execute in devices and IoT gateways with constrained computing (including storage) capabilities. They may also exploit P2P mechanisms to perform their functions in a distributed manner, involving multiple devices. As mentioned at the beginning of this chapter, the combination of Generic Enablers running both in distributed locations and centralized backend systems will allow a scalable, elastic and high performing management of the entire IoT.

IoT Data handling will include GEs dealing with intelligent analysis and mining within massive amounts of data generated by devices and IoT Resources, including both data generated in real-time and historical data. These intelligent analysis/mining will, to a large extent, utilize the BigData Analysis GEs defined in the Data/Context Management chapter and will be based in IoT-specific or application domain (Usage Area) determined logic.

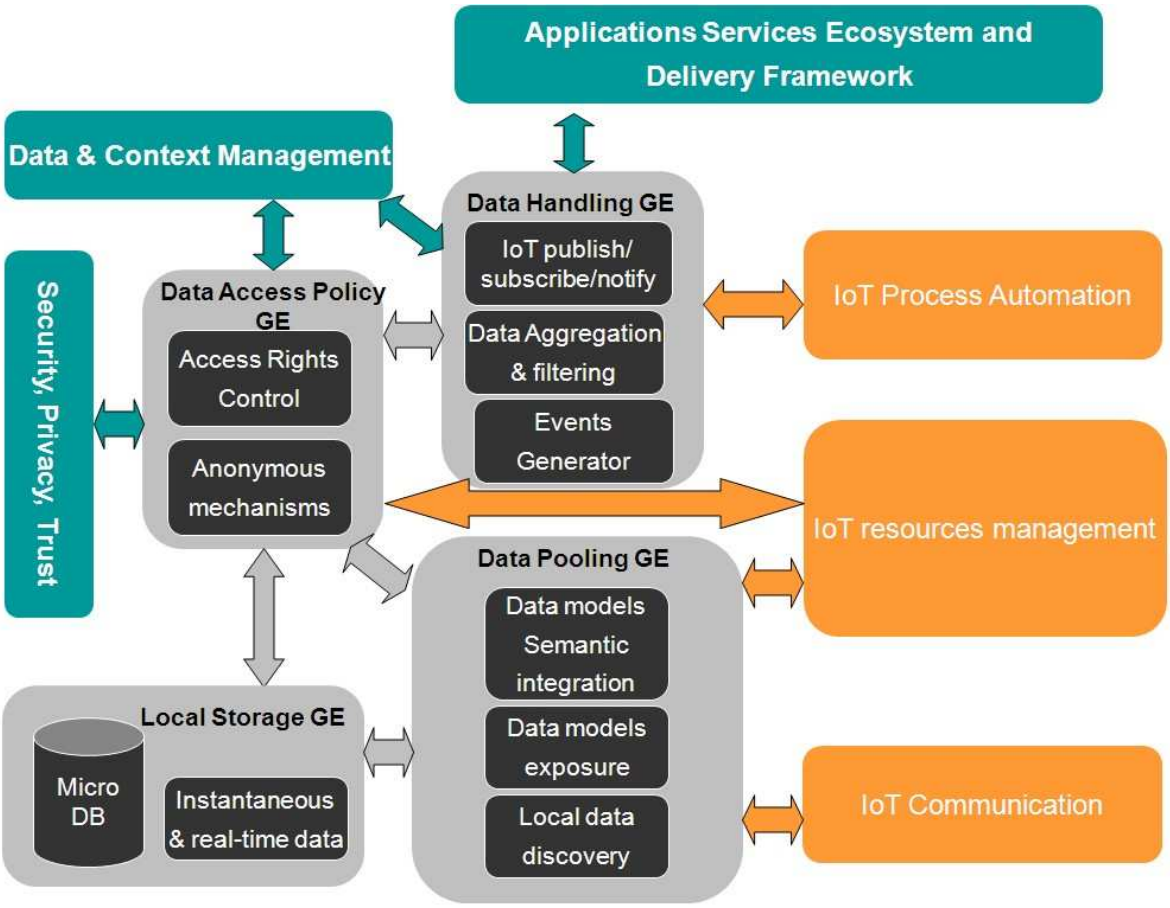


Figure 60 Architecture of IoT Data Handling and its 4 GEs

6.2.3.2 *Description of GEs*

Sheth et al. in [Sheth 08] defined a semantic of sensor Web within Space, Time, and Theme attributes to describe the IoT resources in their respective domain. Barnaghi et al in [Barnaghi 10] use the concepts from the semantic sensor Web and provide a platform for sensor data description and publication using linked-data technologies. In a related work, Wang et al [Wang 09] describe a model for annotation and reasoning over sensor data and discuss possible scenarios for interpretation of this data using linked-data concepts. The SENSEI project also demonstrates the core functions for storing, querying and managing distributed IoT resource description data [SENSEI 10]. A general survey on the IoT that describes its different aspects, components and layers as well as data integrity issues is also provided in [Atzori 10].

IoT Service Enablement will extend the current IoT description models and will create an integrated and interoperable solution for IoT data handling. It will also provide solutions for processing of the resources information about real world and IoT knowledge.

It is expected that things will produce large amount of data in different contexts, depending of energy consumption constraints, privacy rules or nomadic unforeseen turn of events. The IoT Data Handling GEs will provide functions to be able to manage three types of data:

- instantaneous real-time data, which are not relevant to store but could be useful for dedicated applications at a specific time
- locally stored data for privacy and security concerns or due to unreliable and unstable hazardous connectivity across networks, sometimes derived from energy or motion constraints
- globally stored data, which will be stored and provided by any provider for access to elementary or historical data.

For privacy and security concerns, access and storage rights should be defined at the closer level of data production, to devices or to IoT resources. As an example, the Data Access Policy GE is not included in the Security, Privacy, Trust technical chapter which is a more centralized chapter. Data Access Policy GE is a dedicated additional component to the Security, Privacy and Trust since it comprises functionality inside the device/IoT gateway supported by Security, Privacy and Trust which does not have components running at this level of data production, close to the devices or IoT resources.

As IoT would target highly heterogeneous discontinuously connected devices (sensors, actuators or personal mobile terminals) IoT Data Handling would also deal with the different data-models related to the different technologies and protocols in order to provide an homogeneous access to all relevant data for Future Internet applications. IoT Communications would rely on this capability to retrieve the technology and protocol-related data models, providing on its end the homogeneous access to the devices, in other words, hiding the heterogeneity of the devices from the Future Internet applications.

The data models will be identified through different Usage Area requirements and based on previous work already described above. Thus, a common data model will be provided at device, gateway and IoT Backend level to support semantic integration of Future Internet Applications and Services. This data model will be consistent with the general data model defined in FI-WARE as described in the Data/Context Management chapter.

Some data could be stored locally in micro-database for devices or IoT resources with energy constraints in order to limit the number of synchronization or relationships with global storage system. This micro-database is also part of nomadic devices or gateways expected to overcome connectivity problems or emphasize local reactivity.

Access rights features will be manageable by IoT Resources management regarding both Applications policy rules and devices and IoT Resources rules. These rules could be associated to a device or IoT resource profile to enhance privacy management and to provide human and context based changes.

Each device or IoT Resource could change its policy rules, for privacy reasons, especially to provide anonymous or personal data. Security, privacy and Trust Generic Enablers will support IoT Data Handling to define what kind of anonymous mechanisms could be put in place to protect privacy and provide relevant information for applications.

Data flow processing will be supported in order to direct the relevant data from its sources to their respective destinations. It has to be scalable, elastic and high performing distributed across devices, IoT Resources and Gateways. Devices, linked to IoT Resources and gateways, will be able to manage local data and process data based on pattern-condition-action based on IoT-tailored Complex Event Processing Generic Enablers running in the devices.

IoT Data Handling will also implement mechanisms to deliver data in P2P mode between devices and things and to support a more efficient communication of data between IoT resources and applications. IoT-oriented Publish/Subscribe Broker GEs supporting this feature will interwork with centralized Publish/Subscribe Broker GEs both supporting the interfaces defined in the Data/Context Management chapter. This way, it will be feasible to create a reliable network of Publish/Subscribe Brokers that propagates subscription service conditions down to IoT resources.

A proper utilization of GEs both in centralized servers and in devices closer to IoT resources will make it feasible to satisfy the challenging requirements of scale and real-time response linked to IoT Data Handling.

6.2.3.3 *Critical product attributes*

- Flexible data handling including support for various data models, data transformations, data mediations and semantic integrations
- Distributed data management including local storage and processing as well as mechanisms to easily handle the massive amount of IoT data, online stream processing with real-time support and offline processing of collected information based on manageable security rules
- Global definition, real-time access and/or storage of distributed IoT events

6.2.4 IoT Process Automation

6.2.4.1 *Target usage*

IoT Process Automation will propose to Application/Services Providers generic capabilities enabling to use subscription and rules templates that will ease programming of automatic processes involving IoT resources. They will allow to setup high-level conditions that may i.e. trigger new actions in a process.

The distributed nature of the IoT architecture, including mobility of things, requires new forms of business integration and process automation with several levels of collaboration, for example, between devices, gateways and/or components at application level. This may involve, among other aspects, the need to execute part of the process on distributed nodes (e.g., IoT gateways) near or adjacent to IoT resources.

A necessary step for improving IoT processes in large scale solutions is the ability to combine data generated by IoT with business processes that are responsible for scheduling and control. For IoT scenarios, it is necessary to devise a reasonable and complete modelling construct meaning that the process interactions, which may include asynchrony and concurrency, are correct with respect to runtime input-output pairs and that all possible intra- and inter-process interactions are captured adequately by the modelling construct.

Based on semantic capabilities as well as distributed or P2P mechanisms, IoT Process Automation GEs will complement the capabilities and extend the scope of Backend Service Compositions and Front-end Mashup GEs defined in the Applications/Service delivery chapter in FI-WARE. They will allow that the intelligence in Applications/Services dealing with the Future IoT will be based not only on centralized but also on local as well as on distributed processing and decisions executed near or adjacent to IoT resources. These local and distributed processing and decisions will improve reactivity in lots of processes and require new approach for Business Process modelling

6.2.4.2 *Description of GEs*

Process automation is dealing with low level processes describing the interactions of the IoT resources hosted by devices and IoT gateways (micro business rules), providing modelling constructs to describe these interactions, templates to capture the events occurring at this level of interaction and knowledge accumulation from the observation of interaction patterns occurring between the devices and IoT gateways. More specifically, the following features were identified:

- IoT knowledge management: increasing the intelligence of IoT Services capabilities over a long period of time, including

- handler of application domain ontologies,
- knowledge base collector
- a design reasoner to execute classification of assertions and other semantic queries, target a collaborative framework between micro business rules processing engines

- Support to IoT-aware Business Process Management: enabling the programming of Business Processes where part of the process is able to run near or adjacent to IoT resources and gateways. This may imply defining means supporting important IoT characteristics directly into business processes notations.

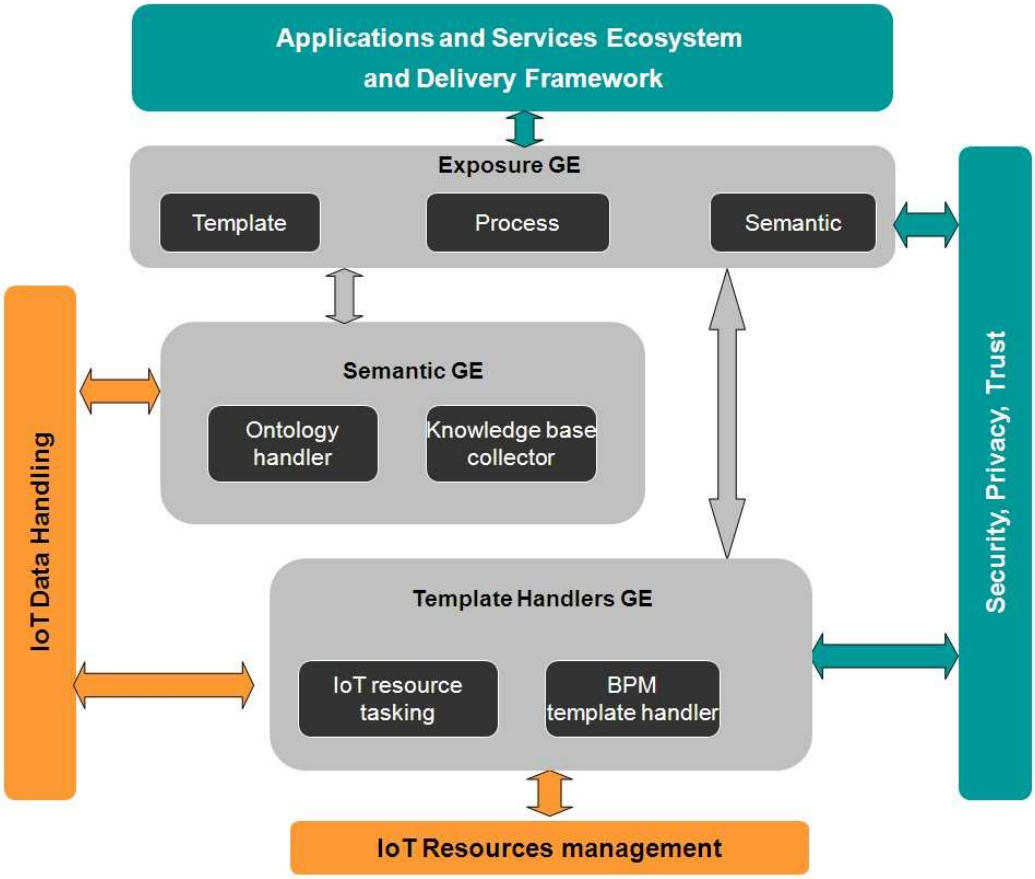


Figure 61 Architecture of IoT Process Automation and its 3 GEs

The Process Automation sub-chapter is organized in 3 GEs, which are the Template Handler GE, the Semantic GE and the Exposure GE.

Data events could be produced and stored locally and depending on local rules could trigger Usage Area processes or applications.

- The functionality of the Template Handler GE is to provide the means to define templates that IoT-oriented GEs running in proximity to IoT resources could take into use. The following components would be considered as part of this GE:
  - The IoT resource tasking component that will allow services to perform request operations from things to the devices
  - The BPM template handler that will provide means to create and to maintain templates for the definition of low level business processes, which devices should execute among themselves without the need to be always connected to the IoT backend. This component will also interact with the Knowledge Base Collector/Accessor in order to detect interaction patterns and, as a result, to define new templates for the future interactions. As an example, a certain pattern of intrusion might be

detected at the level of the Knowledge Base Collector/Accessor. This would results in a recommendation from the BPM Template Handler to define/apply a new low level business process to avoid this new detected and learned intrusion.

- The data event subscription/processing template handler that will provide means to create and maintain templates for the definition of low level subscription queries or complex event processing rules to be submitted to IoT-oriented Publish/Subscribe and Complex Event Processing GEs, respectively.

- The Semantic GE implements functionality through the following components:
  - The knowledge base collector that will collect long-term knowledge from the IoT resources, relying on the functionality provided by IoT Data Handling GEs and trigger (or propose triggering)applications or business processes in real-time.
  - The ontology component that will enable various Usage Areas to deploy their respective domain ontologies on the IoT subsystem.
- The Exposure GE includes:
  - The Template Exposer that will expose the structure and functionality (template) being provided with atemplate handler.
  - The process exposer that will expose the characteristics and operational capabilities of the IoT Resources and Things.
  - The semantic exposer that will expose interfaces towards the ontology handler and towards/from the knowledge base collector

Critical product attributes

- Templates, methodology, modeling tools and support for the orchestrated execution of IoT business processes synchronized with application level business processes using IoT information and triggered by distributed real-time events
- Knowledge generation from long-term events observation at the low level of devices and IoT resources

6.3 Question Marks

6.3.1.1 *Security issues*

Issue 1

Anonymous mechanisms for Data Handling, especially to define what kind of data could be stored for medium and long time following privacy rules defined by „Security, trust, Privacy" technical chapter and provided as requirements by usage area projects

Issue 2

Access security policy control for the management of privacy and security aspects of the IoT resources

Issue 3

IoT Security & Protection schemes for the security, privacy, identity and access management challenges of the IoT process automation function. It provides policies for design or runtime configuration, policy enforcement mechanisms and access authentication scheme from/towards IoT resources and elementary applications hosted by things. Configurable role-management and multi-tenant scenarios are supported as well.

6.3.1.2 *IoT Data as a Service*

While IoT focuses on delivery of data to and from the devices and the application layer, often data needs to be stored and later accessed by applications and / or by the IoT Services for its proper operation and management of IoT resources. This storage should not follow only structured approach but has to support consistency during time-to-time synchronization or sporadic events and provide "Data as a Service" depending on the place, the type and access rights to the relevant events and data demanded by applications. This topic should be addressed in a coordinated fashion together with functionality inside Data and Context Management.

Once data or events are collected and stored, the use of analytics tools could require lots of computational resources. IoT-oriented Analytics relying on the global BigData Analysis GE defined within the Data/Context Management chapter will be applied to historical data with potential additional real-time data. It needs further clarification how the results get applied to IoT and which GEs are influenced.

6.4 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., Usage Area projects in the EU FP7 Future Internet PPP)

The following figure describes the relationships between the concepts defined in this section.

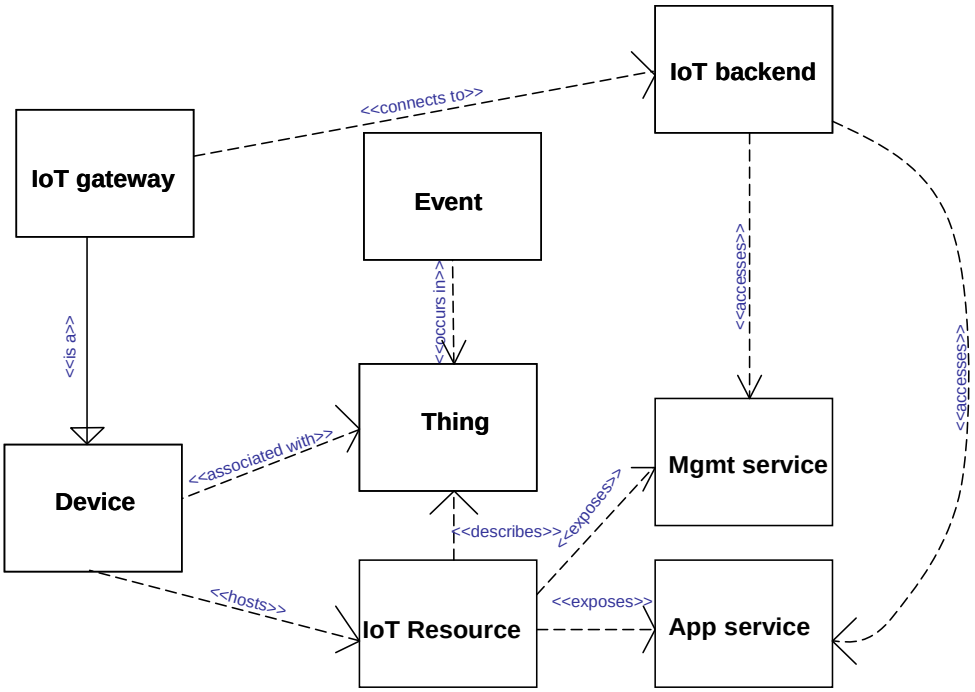


Figure 62 Concepts defined within the IoT Service Enablement technical chapter

- **Thing**. One instance of a physical object, living organism, person or concept interesting to a person or an application from the perspective of a FI-WARE instance, one particular usage area or to several usage areas. Examples of physical objects are: building, table, bridge (classes), the Kreml, the tennis table from John's garden, the Tower bridge (instances). Examples of living organisms are: frog, tree, person (classes), the green frog from the garden, the oak tree in front of this building, Dr. Green.
- **Class of thing**. Defines the type of a thing in the style of object-oriented programming: a construct that is used as a blueprint to create instances, defining constituent members which enable class instances to have state and behavior. An example of class of thing is "person", whereas an instance of a thing is "Dr. Green", with all the static/dynamically changing properties of this particular person.
- **Group of things**. A physical/logical group of things. Examples are all office buildings from Munich, all bridges above the Thames, all screws from a drawer, the planes of Lufthansa currently above Germany, all cars inside a traffic jam driven by a female driver, the trees from the Amazonian forest, the squids from the Mediterranean see the mushrooms from Alice's garden, all the fish from the aquarium of John., the soccer team of Manchester, the colleagues from OrangeLabs currently working *abroad* (from the perspective of France), all patients above 60 years of age of Dr. Green, the traffic jams from Budapest, the wheat crop in France in the year 2011, the Research and Development Department of the company Telefónica.
- **Device**. Hardware entity, component or system that may be in a relationship with a *thing* or a *group of things* called *association*. A device has the means either to measure properties of a thing/group of things and convert it to an analog or digital signal that can be read by a program or user or to potentially influence the properties of a thing/group of things or both to measure/influence. In case when it can only measure the properties, then we call it a sensor. In case when it can potentially influence the properties of a thing, we call it an actuator. Sensors and actuators may be elementary or composed from a set of elementary sensors/actuators. The simplest elementary sensor is a piece of hardware measuring a simple quantity, such as speed of the wind, and displaying it in a mechanical fashion. Sensors may be the combination of software and hardware components, such as a vehicle on-board unit, that consists of simple sensors measuring various quantities such as the speed of the car, fuel consumption etc and a wireless module transmitting the measurement result to an application platform. The simplest elementary actuator is a light switch. We have emphasized *potentially* because as a result of the light switch being switched on it is not sure that the effect will be that light bulb goes on: only an *associated* sensor will be able to determine whether it went on or not. Other examples of devices are smart meters, mobile POS devices. More sophisticated devices may have a unique identifier, embedded computing capabilities and local data storage utilities, as well as embedded communication capabilities over short and/or long distances to communicate with IoT backend. Simple devices may not have all these capabilities and they can for instance only disclose the measurement results via mechanical means. In this latter case the further disclosure of the measurement result towards IoT backend requires another kind of device, called IoT Gateway.
- **Association**. It is a physical/logical relationship between a device and a thing or a device and a group of things or a group of devices and a group of things from the perspective of a FI-WARE instance, an application within a usage area project or other stakeholder. A device is associated with a thing if it can sense or (potentially) influence at least one property of the thing, property capturing an aspect of the thing interesting to a person or an application from the perspective of a FI-WARE instance, one particular usage area or to several usage areas. Devices are associated with things in *fully dynamic* or *mainly static* or *fully static* manner. The association may have several different embodiments: *physical attachment*, *physical embedding*, *physical neighborhood*, *logical relation* etc. Physical attachment means that the device is physically attached to the thing in order to monitor and interact with it, enabling the thing to be connected to the Internet. An example is the on-board device installed inside the vehicle to allow sending sensor data from the car to the FI-WARE instances. Physical embedding means that the device is deeply built inside the thing or almost part of the thing. An example of physical embedding is the GPS sensor inside a mobile phone. Physical neighborhood means that the device is in the physical neighborhood of the thing, but not in physical contact with it. An example of

physical neighborhood is the association of the mobile phone of a subscriber who is caught in a traffic jam with the traffic jam itself: the mobile phone is the device, the traffic jam is the thing and the association means that the mobile phone is in the physical neighborhood of the area inside which the traffic jam is constrained (e.g. within 100 m from the region where the average speed of cars is below 5 km/h). Logical association means that there is a relationship between the device and the thing which is neither fully physical in the sense of attachment or embedding, nor fully physical proximity related. An example of logical association is between the car and the garage door opener of the garage where the car usually parks during the night.

- **IoT gateway**. A device that additionally to or instead of sensing/actuating provides inter-networking and protocol conversion functionalities between devices and IoT backend potentially in any combination of these hosts a number of features of one or several Generic Enablers of the IoT Service Enablement. It is usually located at proximity of the devices to be connected. An example of an IoT gateway is a home gateway that may represent an aggregation point for all the sensors/actuators inside a smart home. The IoT gateway will support all the IoT backend features, taking into consideration the local constraints of devices such as the available computing, power, storage and energy consumption. The level of functional split between the IoT backend and the IoT gateway will also depend on the available resources on the IoT gateway, the cost and quality of connectivity and the desired level for the distribution of intelligence and service abstraction.
- **IoT resource**. Computational elements (software) that provide the technical means to perform sensing and/or actuation on the device. There may be one-to-one or one-to-many relationship between a device and its IoT resource. Actuation capabilities exposed by the IoT resource may comprise configuration of the management/application features of the device, such as connectivity, access control, information, while sensing may comprise the gathering of faults, performance metrics, accounting/administration data from the device, as well as application data about the properties of the thing with which the device is associated. The resource is usually hosted on the device.
- **Management service**. It is the feature of the IoT resource providing programmatic access to readable and/or writable data belonging to the functioning of the device, comprising a subset of the FCAPS categories, that is, configuration management, fault management, accounting /access management/administration, performance management/provisioning and security management. The extent of the subset depends on the usage area. Example of configuration management data is the IP endpoint of the IoT backend instance to which the device communicates. Example of fault management is an error given as a result of the overheating of the device because of extensive exposure to the sun. Example of accounting/administration is setting the link quota for applications using data from a certain sensor. Example of security management is the provisioning of a list of device ID-s of peer devices with which the device may directly communicate.
- **Application service**. It is the feature of the IoT resource providing programmatic access to readable or writable data in connection with the thing which is associated with the device hosting the resource. The application service exchanges application data with another device (including IoT gateway) and/or the IoT backend. Measured sensory data are example of application data flowing from devices to sensors. Setting the steering angle of a security camera or sending a “start wetting” command to the irrigation system are examples of application data flowing from the application towards the sensor.
- **Event**. An event can be defined as an activity that happens, occurs in a device, gateway, IoT backend or is created by a software component inside the IoT service enablement. The digital representation of this activity in a device, IoT resource, IoT gateway or IoT backend instance, or more generally, in a FI-WARE instance, is also called an event. Events may be simple and complex. A simple event is an event that is not an abstraction or composition of other events. An example of a simple event is that “smart meter X got broken”. A complex event is an abstraction of other events called member events. For example a stock market crash or a cellular network blackout is an abstraction denoting many thousand of member events. In many cases a complex event references the set of its members, the implication being that the event contains a reference. For example, the cellular network blackout may be caused by a power failure of the air conditioning in the operator’s data center. In other cases such a reference may not exist. For example there is no accepted agreement as to which events are members of a stock

market crash complex event. Complex events may be created of simple events or other complex events by an IoT resource or the IoT backend instance.

- **IoT Backend.** Provides management functionalities for the devices and IoT domain-specific support for the applications. Integrant component of FI-WARE instances.
- **IoT application**. An application that uses the application programming interface of the IoT service enablement component. May have parts running on one/set of devices, one/set of IoT gateways and one/set of IoT backends.
- **Virtual thing**. It is the digital representation of a thing inside the IoT service enablement component. Consists of a set of properties which are interesting to a person or an application from the perspective of a FI-WARE instance, one particular usage area or to several usage areas. Classes of things have the same set of properties, only the value of the properties change from instance to instance.

6.5 References

[Harbour 10]	Harbour Research, Machine-to-Machine & Smart System Forecast, 2010-2014, http://www.harborresearch.com/HarborContent/m2msmartsystems.html
[IoT Brussels 09]	Internet of Things — An action plan for Europe Brussels, 18.6.2009 COM(2009), http://www.google.com/url?sa=t&source=web&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fwww.kowi.de%2FPortaldata%2F2%2FResources%2Ffp7%2Fcom-2009-278.pdf&ei=mtT5TeOJJYumvgOjyZSyAw&usg=AFQjCNEb1JOgZj0ZbKTsxVpniUMr4W2wzQ
[Juniper 11]	Juniper Networks, Machine-to-machine (M2M) – The Rise of the Machines, 2011, http://www.google.com/url?sa=t&source=web&cd=6&ved=0CEQQFjAF&url=http%3A%2F%2Fwww.juniper.net%2Fus%2Fen%2Flocal%2Fpdf%2Fwhitepapers%2F2000416-en.pdf&ei=TtX5TdbLH4_uuAOJ_eyPAw&usg=AFQjCNGg3FIyQ79SA5DJJXfC_rREKWt7gw
[Prabhu 06]	B. S. Prabhu, X. Su, H. Ramamurthy, C-C. Chu and R. Gadh (2006) “Winrfid, a middleware for the enablement of Radio Frequency Identification (RFID) based Applications”, Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions (Wiley, 2006).
[IEEE802.15.4]	I. C. Society (2006) "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)". IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003), 2006.
[6LoWPan]	IETF 6lowpan Working Group: http://www.ietf.org/html.charters/6lowpan-charter.html
[Abangar 10]	H. Abangar et al (2010) “A service oriented middleware network architecture for wireless sensor networks”, Future Network & Mobile Summit 2010
[Hadim 06]	S. Hadim and N. Mohamed (2006) “Middleware Challenges and Approaches for Wireless Sensor Networks”. IEEE Distributed Systems Online, vol. 7, no. 3, 2006.
[Heinzelman 04]	W.B Heinzelman, A.L. Murphy, H.S. Carvalho and M.A. Perillo (2004) "Middleware to support sensor network applications". IEEE Network, vol.18, no.1, pp. 6-14, 2004.

[ETSI-M2M]	ETSI. (2010) "Machine to Machine Communications". http://www.etsi.org/Website/Technologies/M2M.aspx
[Thiesse 09]	F. Thiesse, C.Flörkemeier, M. Harrison, F. Michahelles and C. Roduner (2009) "Technology, Standards, and Real-World Deployments of the EPC Networks". IEEE Internet Computing 13 (2), pp. 36-42, 2009.
[ANSI 06]	Data format standard for radiation detectors used for homeland security, http://physics.nist.gov/Divisions/Div846/Gp4/ANSIN4242/xml.html
[CCSI 09]	Common CBRN Sensor Interface (CCSI) (2009) www.jpeocbd.osd.mil
[ISOX73]	IEEE (2004) "ISO/IEEE11073 (X73), Health informatics - Point-of-care medical device communication". Retrieved from http://www.iso.org
[ZigbeeAll]	ZigBee Alliance (2010). http:// www.zigbee.org
IEEE1451	Lee, K. (2000) "IEEE 1451: A standarad in support of smart transducer networking". In Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, 2000. IMTC 2000, pp. 525-528. Home page: http://ieee1451.nist.gov/
[OGC 09]	OGC OWS-6 Geoprocessing Workflow Architecture Engineering Report, http://www.google.com/url?sa=t&source=web&cd=2&ved=0CCIQFjAB&url=http%3A%2F%2Fportal.opengeospatial.org%2Ffiles%2F%3Fartifact_id%3D34968&ei=9DLaTZvmBYGFtgeC3KjpDg&usg=AFQjCNHo-ajNYtMmgH1bHNBEisKL69qD9w
[Botts 08]	Botts, M., Percivall, G., Reed, C., & Davidson, J. (2008). "OGC® Sensor Web Enablement: Overview and High Level Architecture". In F. Fiedrich & B. Van De Walle (Eds.), GeoSensor Networks (Vol. 4540, p. 175-190). Springer. Retrieved from http://portal.opengeospatial.org/files/?artifact_id=25562
[SensorML]	SensorML, Sensor Model Language (SensorML), The Open Geospatial Consortium, http://www.opengeospatial.org/standards/sensorml/
[Russomanno 05]	D. J. Russomanno, C. Kothari, and O. Thomas (2005) “Sensor ontologies: from shallow to deep models," in Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, pp. 107-112, 2005.
[Kim 08]	J. H. Kim, K. Kwon, H., K. D.-H., H.-Y., S. J. Lee (2008), “Building a service-oriented ontology for wireless sensor networks," in Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science, pp. 649-654, 2008.
[Barnaghi 09]	P. M. Barnaghi, S. Meissner, M. Presser, and K. Moessner (2009) “Sense and Extensible: Semantic Data Modelling for Sensor Networks”, In Proc. Of the ICT-Mobile Summit, June 2009.
[W3CSSN-XG 10]	http://www.w3.org/2005/Incubator/ssn/wiki/
[Anyanwu 03]	K. Anyanwu, A. P. Sheth, (2003) “p-Queries: Enabling Querying for Semantic Associations on theSemantic Web”, In Proceedings of WWW 2003 Conference, pp. 690–699, 2003.
[Ding 04]	L. Ding, T Finin, A Joshi, R Pan, R S Cost, Y Peng, P Reddivari, V Doshi and J Sachs,

	Swoogle: a search and metadata engine for the semantic web. CIKM ’04, pp. 652–659, 2004.
[Wang 08]	M. Wang, J. Cao, J. Li, and S. k. Dasi, (2008), "Middleware for Wireless Sensor Networks: A Survey," Journal of Computer Science and Technology, vol. 23, no. 3, pp. 305-326, May 2008.
[UDDI 04]	OASIS Open (2004) "UDDI Version 3.0.2". http://www.uddi.org/pubs/uddi_v3.htm
[Jianjun 07]	Y. Jianjun, G. Shengmin, S. Hao, Z. Hui and X. Ke (2007) “A kernel based structure matching for web services search”. In Proceedings of the 16th international Conference on World Wide Web (WWW '07), pp. 1249-1250, 2007.
[Willmott 05]	S. Willmott, H. Ronsdorf, K. H. Krempels (2005) “Publish and Search versus Registries for Semantic Web Service Discovery”. In Proceedings of the 2005 IEEE/WIC/ACM international Conference on Web intelligence, pp. 491-494, 2005.
[Platzer 09]	C. Platzer, F. Rosenberg and S. Dustdar (2009) “Web service clustering using multidimensional angles as proximity measures”, ACM Trans. Internet Technol. 9, 3, pp. 1 26, 2009.
[Toch 07]	E. Toch, A. Gal, I. Reinhartz-Berger, D. Dori (2007), “A semantic approach to approximate service retrieval”, ACM Trans. Internet Technol. 8, 1, 2007.
[Ma 08]	J. Ma, Y. Zhang, J. He, (2008) “Efficiently finding web services using a clustering semantic approach”, In Proceedings of the 2008 international Workshop on Context Enabled Source and Service Selection, integration and Adaptation: Organized with the 17th international World Wide Web Conference (WWW 2008), pp. 1-8, 2008.
[Elahi 09]	B. M. Elahi, K. Römer, B. Ostermaier, M. Fahrmair and W. Kellerer (2009) “Sensor ranking: A primitive for efficient content-based sensor search”, In Proceedings of the 2009 international Conference on information Processing in Sensor Networks, pp. 217-228, 2009.
[Ostermaier 08]	B. Ostermaier, B. M. Elahi, K. Römer, M. Fahrmair and W. Kellerer (2008) “Dyser: towards a real-time search engine for the web of things”, In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 429-430, 2008.
[Autonomic]	Jeffrey O. Kephart and David M. Chess (2003) "Autonomic Manifesto", IBM Thomas J, Watson Research Center. Retrieved of http://www.research.ibm.com/autonomic/manifesto
[IBM 03]	International Business Machines Corp (2003) “An Architectural Blueprint for Autonomic Computing”, April 2003.
[Kephart 03]	Kephart, J. O., & Chess, D. M. (2003). "The vision of autonomic computing". Computer, 36(1), 41-50. IEEE Computer Society Press. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1160055
[Fehskens 89]	A. Fehskens (1989) “Monitoring Systems” 1st IFIP Integrated Network Management Symposium, Boston, May 1989
[Strassner 04]	J. Strassner (2004), “Autonomic Networking – Theory and Practice”, IEEE Tutorial, Dec 2004.

[Raz 00]	D. Raz and Y. Shavitt (2000) “Active Networks for Efficient Distributed Network Management”, IEEE Communications Magazine, Vol. 38, No. 3, pp. 138-143, 2000.
[Gu 05]	T. Gu, X. Wang, H. Pung, and D. Zhang (2005) "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications, no. 28, pp. 1-18, 2005.
[Sheth 08]	A. Sheth, C. Henson and S. Sahoo (2008) “Semantic sensor web", Internet Computing, IEEE, vol. 12, no. 4, pp. 78-83, July-Aug. 2008.
[Barnaghi 10]	P. Barnaghi, M. Presser, K. Moessner (2010) "Publishing Linked Sensor Data"', In Proc. of the 3rd Int. Workshop on Semantic Sensor Networks (SSN), Nov. 2010.
[Wang 09]	W. Wang and P. M. Barnaghi (2009) "Semantic Annotation and Reasoning for Sensor Data". In Proc. of the 4th European Conference on Smart Sensing and Context (EuroSSC2009), LNSC, Springer, Sep. 2009.
[SENSEI 10]	Sensei Consortium (2010) "Sensei Project, the Sensei Real World Internet Architecture", in http://www.sensei-project.eu/, white paper.
[Atzori 10]	L. Atzori, A. Iera, G. Morabito (2010), “The Internet of Things: A survey”, Computer Networks, vol 54, issue 15, pp. 2787-2805, 2010.
[Palmer 03]	N. Palmer (2003) “BPM 2003 Market Milestone Report”. A Delphi Group White Paper, 2003.
[Curtis 92]	W. Curtis, M. I. Kellner, and J. Over (1992) “Process modelling”. Commun. ACM, 35(9):75-90, 1992. ISSN0001-0782.
[Scheer 01]	A.W. Scheer (2001) “ARIS - Modellierungsmethoden, Metamodelle, Anwendungen”. Springer, 2001. ISBN 3-540-41601-3.
[Gschwind 08]	T. Gschwind, J. Koehler, and J. Wong (2008) “Applying patterns during business process modelling”. Business Process Management, volume 5240 of Lecture Notes in Computer Science, pages 4-19. Springer, 2008. ISBN 978-3-540-85757-0.
[Koehler 07]	J. Koehler and J. Vanhatalo (2007). “Process anti-patterns: How to avoid the common traps of business process modeling, part 1 and 2”. IBM WebSphere Developer Technical Journal, issue 10.2, February 2007.
[Preist 04]	C. Preist (2004) “A Conceptual Architecture for Semantic Web Services”. In Proceedings of the 3rd International Semantic Web Conference (ISWC), 2004.
[Toma 05]	I. Toma, K. Iqbal, M. Moran, D. Roman, T. Strang, and D. Fensel (2005) “An Evaluation of Discovery approaches in Grid and Web services Environments”. In Proceedings of NODe/GSEM, 2005.
[Meyer 06]	H. Meyer and M. Weske (2006) “Automated service composition using heuristic search”. In Proceedings of the Fourth International Conference on Business Process Management (BPM 2006), pp. 81-96, 2006.
[Born 08a]	M. Born, A. Filipowska, M. Kaczmarek, and I. Markovic. (2008) “Business functions ontology and its application in semantic business process modelling”. In Proceedings of the 19th Australasian Conference on Information Systems, Christchurch, New Zealand,

	Dec 2008.
[Weske 07]	M. Weske (2007). “Business Process Management: Concepts, Languages, Architectures”. Springer, 2007. ISBN 978-3-540-73521-2.
[List 06]	B. List and B. Korherr (2006) “An evaluation of conceptual business process modelling languages”. In proceedings of the 2006 ACM symposium on Applied computing (SAC'06), pages 1532_1539, New York, NY, USA, 2006. ACM. ISBN 1-59593-108-2.
[Hadar 06]	Hadar, I., & Soffer, P. (2006), "Variations in conceptual modeling: classification and ontological analysis". Journal of the Association for Information Systems, 7(8), 568-592. Retrieved from http://ais.bepress.com/cgi/viewcontent.cgi?article=1270&context=jais
[Spiess 08]	P. Spiess, D. Khoa Nguyen and I. Weber and I. Markovic and M. Beigl (2008) ”Modelling, Simulation, and Performance Analysis of Business Processes Involving Ubiquitous Systems". Conference on Advanced Information Systems Engineering (CAiSE), 2008.
[BPMN]	Object Management Group (2011) “Business Process Model and Notation (BPMN) Version 2.0” OMG Standard, January 2011 retrieved at http://www.omg.org/spec/BPMN/2.0/ (accessed May 25, 2011).

7 Interface to Networks and Devices

7.1 Overview

The growing number of heterogeneous devices which can be used to access a variety of physical networks, contents, services, and information provided by a broad range of network, application and service providers has clearly created the conditions required for an increasing number of users to be always in touch with what is going on in the world, both for personal and work-related purposes. Driven by mobility, more and more services are consumed on the move. Therefore mobility is to be provided, and a certain Quality of Experience (QoE) and security are expected. Unfortunately, the complexity of the problem, as well as the lack of standardised interfaces and protocols, makes it difficult to easily and seamlessly support such features at the application layer.

Moreover, in the attempt to differentiate the offer, device manufacturers are continuously introducing new and more sophisticated features in their products. Further, to make the scenario even more fragmented, a variety of development paradigms and technologies are available for different types of connected devices. We can broadly define three programming paradigms as reference for developers interested to deploying their applications on connected devices (and not only for them), i.e. the native programming technologies (Java, Objective-C, C, C++, etc.), the interpreted programming technologies (Java, .NET, Python, etc.) and the script programming technologies (HTML, CSS, Javascript, XML, etc.). The difficulties found in the interoperability among applications running on different devices and the portability of applications across devices, both due to lack of standards and existence of fragmented paradigms, are often a big issue in the development of global Internet Applications. FI-WARE will address the definition of Generic Enablers implementing a common and standard **Interface to Devices** that helps to minimize these limitations.

As a further element to be addressed, in the past communication solutions were mainly developed according to their usage. Typical examples are mobile communication, analogue and digital telephony, and Internet. Each of these technologies had their own platforms, environments, and infrastructures. This resulted to silo platforms within the infrastructures of telecommunication providers. With the fixed-mobile-convergence (FMC) strategy, where mobile and fixed and Internet grew together, it was necessary to build and replace the communication technologies by one specific and unique technology and protocol environments. Driven by the economic benefits (relatively cheap network nodes) and the success of Internet end-systems (such as laptops and desktop computers) the packet based technology had its triumphal procession. The Internet is based on a simple transport stack (TCP/IP). To solve the specific application and usage area challenges, overlays were built. More and more the overlay philosophy enabled the clustering of technologies, service and application management, and the control of networks. The Internet had also the philosophy to push the intelligent of networks to the edge, using transport networks as dumb pipes and developing applications/services “over the top”.

FI-WARE considers the **Intelligent Connectivity** at the basis of the Future Internet. The Intelligent Connectivity concept will mean connecting applications and application platforms to the network intelligently, leveraging on the full potential of the features of the network, through the definition of Generic Enablers implementing a standard **Interface to the Networks**. This way, FI-WARE will be capable of exploiting the features of networks offering “smart pipe” functionalities as compare to just be designed to run “over the top”.

The Generic Enablers provided to implement a standardised Interface to Networks and Devices (**I2ND**), can be used by other FI-WARE elements, such as Cloud Hosting, Internet of Things, etc. They can also be directly used by the applications in multiple Usage Areas.

A fundamental challenge for the implementation of the required interfaces is that the network functionality is typically distributed over different elements potentially implemented internally in different ways (in multi-vendor environments), and that the interfaces have to take into account the constraints of different providers (in multi-network service scenarios) as well as legal and regulatory requirements.

For logical reasons, explanations, and implementation issues, a common reference model is needed. The model should reflect the standard communication understanding as well as future and ongoing activities in the field of packet (Internet) based communication, traffic engineering, and service offering. The control and the management of packet (Internet) based networks should be included as well.

A good reference communication model is an extension of the model from the Telemanagement Forum (TMF) [TMF] – the three Strata Model [StrataModel] from the IPsphere Project [IPsphere] is shown in Figure 63. These strata can be seen as overlays; which is in line with the already mentioned overlay philosophy of the Internet.

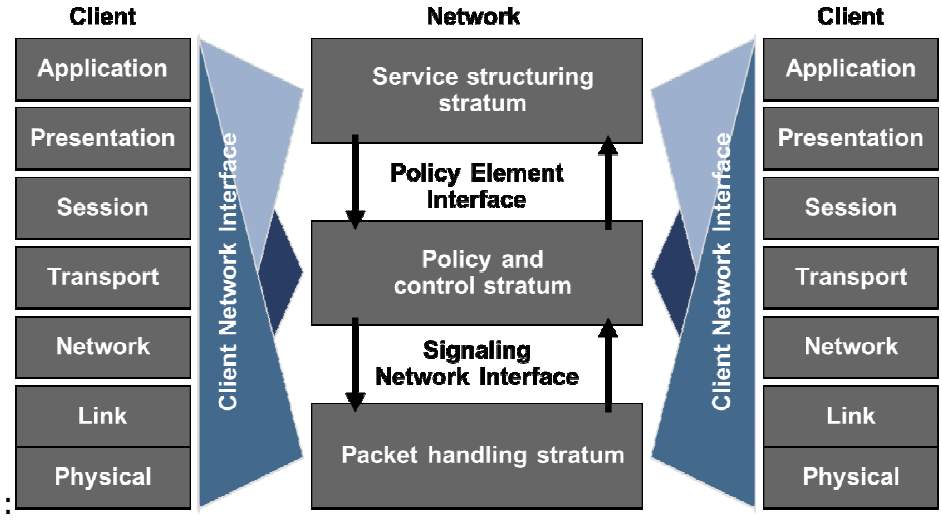


Figure 63 Three strata model for communication in an IP driven operator infrastructure

TMF, within the Service-structuring stratum, addresses essential requirement of composition of services across multiple stakeholders and multiple technologies. In (TR158_IPsphere_ R2-0 Arch_Doc), the TM Forum IPsphere Framework (IPSF) is introduced, a framework for service providers to service provider Business-to-Business (B2B) interactions and automated collaboration. This IPSF contains main functionalities for service negotiation (Publishing, composition, fulfilment and assurance) that are key aspects for delivering of network services based on Service Level Agreement.

Remark: In the framework of this activity it is planned to use the existing results from IPsphere/TMF and the related EU-R&D-activities, which are in cooperation with IPsphere/TMF. It is also planned to push new – in the framework of FI-WARE developed – results by help of the partners into the TMF for discussion and to shape new releases.

Therefore, all FI-WARE Generic Enablers aiming at the provisioning of services with agreed QoS (like Generic Enablers in the Cloud Hosting chapter) shall need to take into account that 'Negotiation' processes can be put in place with the control stratum of the Networks and Devices infrastructure, using the interfaces being defined in the FI-WARE I2ND chapter. By implementing this negotiation, they will be able to bring a differential value compared to other application platforms merely designed over the top.

In [Cube01] and [Cube02], the authors have extended the Strata Model with three planes: the Data plane, the Control plane, and the Management plane according to Figure 64

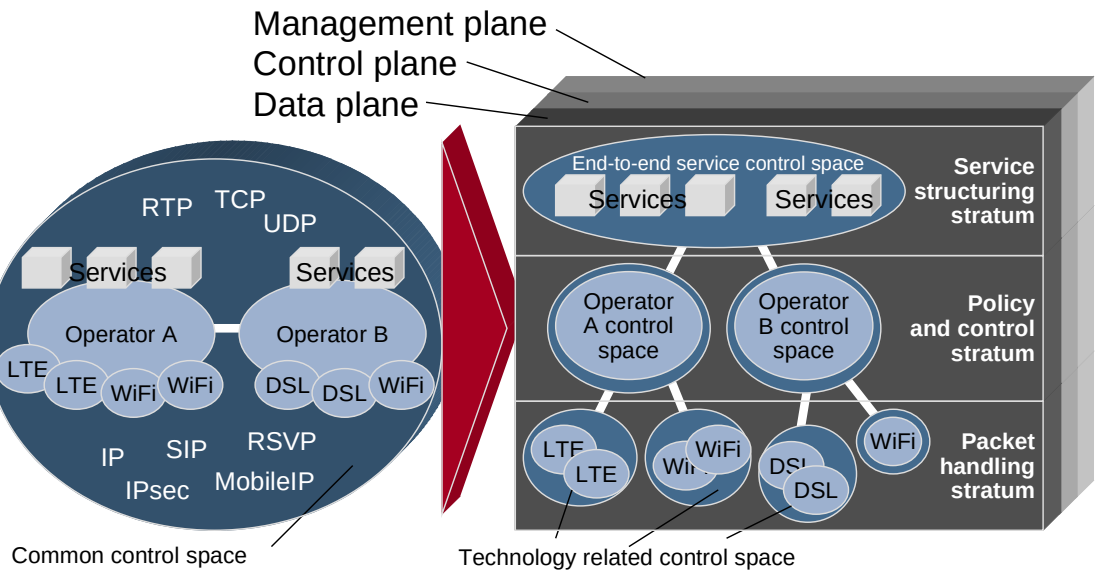


Figure 64 Three strata model for communication extended by NGN principles

The previous figure depicts in the left the current situation in the Internet. Within domains, different interfaces and protocols are implemented to support the operator in the more or less difficult task to run a network efficiently and to give access to end-customer, third party providers, content providers as well as interconnection to other network service providers. On the right side, this is mapped to the proposed communication model, where:

- The Packet Handling Stratum represents the communication technology. Typical examples include access networks such as the 3GPP Long Term Evolution (LTE) or Digital Subscriber Lines (DSL), as well as backbone/core technologies like Carrier Grade Ethernet and other optical network technologies.
- The Control and Policy Stratum is a layer that is responsible for inter-technology issues within operator/network service provider environments. A potential implementation in wireless access networks can be based on the Evolved Packet Core (EPC).
- The Service Structuring Stratum deals with the end-to-end-service provisioning.

The architecture of communication networks can further be separated into three planes: data plane, control plane, and management plane. For illustration and for simplification the following description uses the “three Strata Model” without explicitly distinguishing their respective separation in the data plane, control, and management plane according to the NGN architecture.

7.1.1 I2ND high-level Architecture

The I2ND chapter will address four different classes of interfaces: **connected device, cloud proxy, open networking**, and **network services**. These four functional components directly follow from the four different entities that an interface can refer to: (1) Interfaces to end devices (addressed by connected device), (2) interfaces to gateways (cloud proxy), (3) interfaces to connectivity functions inside the network (open networking), and (4) interfaces to services offered by the network operators (network services).

In each case, the purpose of the interface is both to expose the corresponding network state information to the user of the interface as well as to offer a defined level of control and management (network operation), in order to overcome the limitations of today's network and device interfaces. A control and policy strata/system handles interactions between these different functional components and inter-domain operation.

The four classes of interfaces can be mapped to the Communication Model (without the planes) as depicted in Figure 65

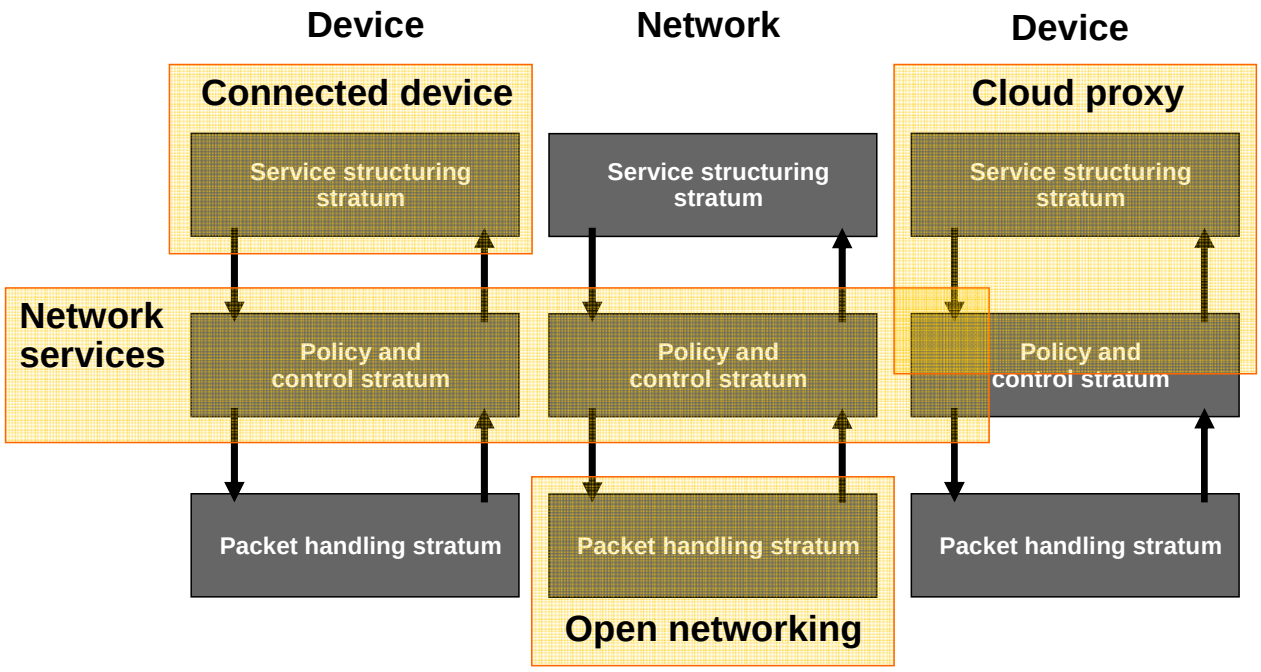


Figure 65 Mapping of the Generic Enablers into the Communication Model

The I2ND chapter addresses the interfaces at the level of different strata. The Service Structuring Stratum includes interfaces towards applications in a wide range of Usage Areas. The chapter will not handle the implementation of the Service Structuring Stratum inside the network. The Packet Handling Stratum in the devices and Cloud proxies, as well as the Policy and Control Stratum in the Cloud Proxies, are well defined and standardised and therefore well established interfaces will be used.

Figure 66 provides a high level view of the Interface to Network and Devices architecture, composed of the following four Generic Enablers:

- **Connected Device Interfacing (CDI)**: this is the Generic Enabler (GE) providing to FI-WARE and Use Case Project applications the possibility to exploit the device features and capabilities, through the implementation of interfaces and related APIs towards the connected devices. The CDI GE interfaces the FI-WARE applications providing unified APIs for app developers. It also interfaces services offered by other FI-WARE GEs, through the internal interface with the S3C Generic Enabler shown as a dotted line in Figure 66 The interfaces provided by CDI GE will reflect the status and the situation of devices and their connected users. Network context information, location data (generated by network nodes) and subsets of the user profiles will be provided in a open, standardised way towards the executing systems. Current standards and discussions in international initiatives and bodies, in particular GSMA (oneAPI), WAC and W3C, will be taken into account, as well as the evolution of device platforms (e.g. MeeGo, Android). The CDI GE, on one hand, will specify interfaces and APIs according to the standardised frameworks mentioned, and will also further improve them, by contributing to the relevant initiatives with proposals for an evolution, which takes into account the Future Internet requirements emerging from FI-WARE and the Use Case Projects.
- **Cloud Edge (CE)**: this is the GE in charge of interfacing the Cloud Proxies with the FI-WARE and legacy cloud services. Users own and use more and more complex home networks connecting many consumer electronic devices and broadband home gateways providing more and more advanced functionalities. The interface and interoperability between all these devices is still a challenge, even after years of development of interoperability standards such as UPNP or DLNA. In the future, gateways will be further extended to specifically include cloud functionality, e. g., in the form of “nano data centres” or “advanced home hubs” that support private cloud functions, execution of downloadable applications in virtualised environments, advanced storage, intelligent content distribution, or translation to local IoT-related networks. This means that home gateways have to be

able to expose interfaces towards the FI-WARE platform to access cloud proxy features, including the interfaces to access objects and devices connected to the cloud proxy, to manage/access local data and to manage devices. The interfaces provided by the CE GE cover a variety of cloud proxies functionalities: end-device communication, home system management, virtual machine management, protocol adaptation, etc. The CE interfaces are also used internally by the S3C Generic Enabler to control the cloud proxies.

- **Network Information & Control (NetIC)**: this is the GE providing a homogeneous access to heterogeneous open networking devices. It exposes network status information and it enables a certain level of programmability within the network, e.g., concerning flow processing, routing, addressing, and resource management at flow and circuit level. Such interfaces also enable network virtualisation, i. e., the abstraction of the physical network resources as well as their control by a virtual network provider. A key advantage of open networking is the implementation of tailored network mechanisms (own addressing schemes or protocols, dynamic tunnels, etc.). Also, premium network services can be implemented, e.g., for the interconnection of private and public clouds by dedicated links with guaranteed Quality-of-Service (QoS). Potential users of NetIC interfaces include network service providers or other components of FI-WARE, such as cloud hosting. Network operators, but also virtual network operators and service providers within the constraints defined by contracts with the operators may access, by means of specific FI-WARE services, the open networks to both set control policies and optimally exploit the network capabilities, and also to retrieve information and statistics about network utilization (e.g., a usage area implementing a smart grid application will rent from an operator resources to build its own virtual network, then this usage area will interface directly with NetIC). This GE also offers a set of I2ND internal interfaces to the S3C in order to control the open networks.
- **Service, Capability, Connectivity and Control (S3C)**: this is the GE providing access to legacy network devices features, capabilities and services. In particular, the aim of this GE is to mitigate the challenges of packet core networks by offering extended interfaces for various service platforms. S3C defines interfaces that are used to control the access to the heterogeneous network infrastructures, and interfaces that provide information from the network management and/or operation support systems, such as auditing data-sets that can be used for billing and charging, or legal interception information that can be used in case of violation of the network environment usage rules. The S3C functional component therefore mainly considers interfaces at service level, whereas the NetIC interface focuses on transport mechanisms. Yet, both functional components may depend on each other, e. g., in case of inter-domain operation, and I2ND will therefore align these interfaces. A central instantiation of the network management is the policy and control system. The targeted policy and control system is a further developed IP Multimedia Subsystem (IMS) and/or Evolved Packet Core (EPC) platform. For inter-connection of network service providers and their domains for example the provisioning of QoS, an inter-carrier sub-enabler will be defined. The enabler will not be accessible directly only through a high level interface for other operators and application developers In addition, it also exposes a set of interfaces to the CE GE that may be used by Cloud Services

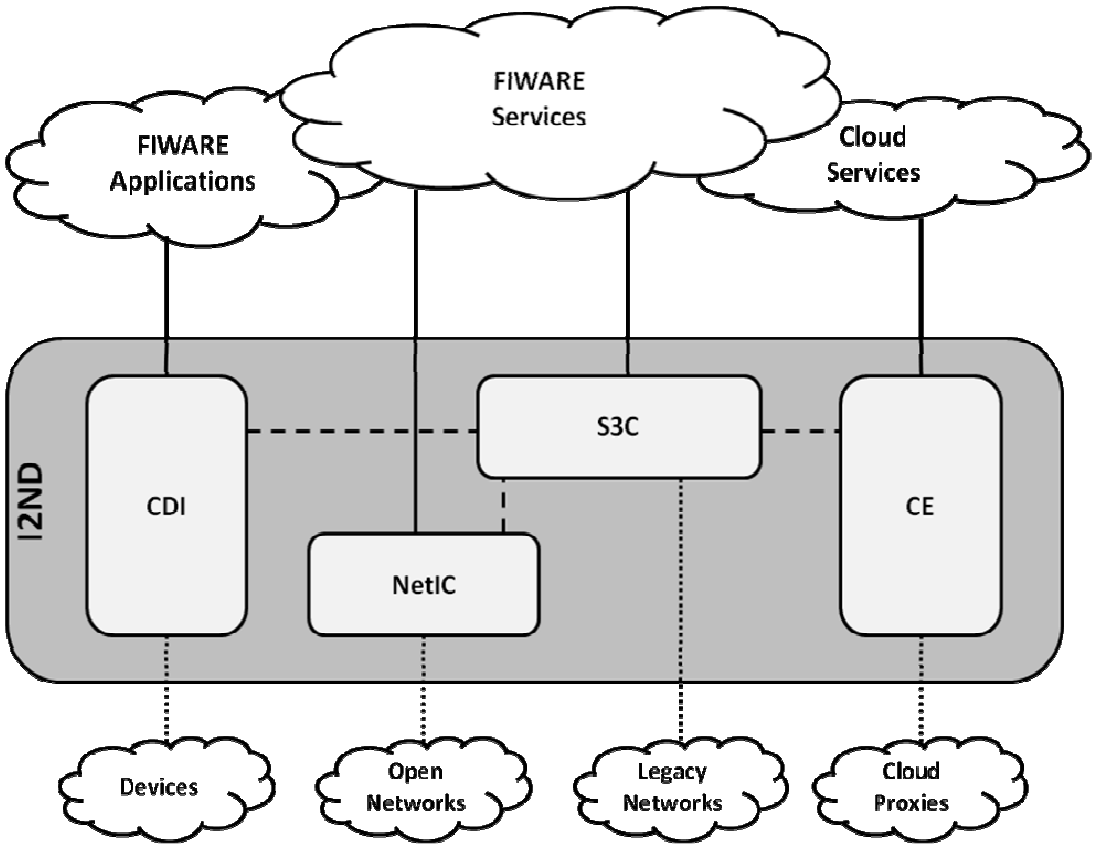


Figure 66 General Architecture of Interface to the Network and Devices (I2ND)

The four I2ND Generic Enablers build the interface between the legacy domain (Devices, Open networks, Legacy network services and Cloud proxies) and the rest of the FI-WARE domain (applications, services and clouds) as illustrated in Figure 66. The dashed lines shown in Figure 66 represent the internal (within the I2ND scope) interfaces; the dotted lines represent the interactions between the I2ND Generic Enablers and the underlying legacy domain elements; the solid lines represent the interfaces between the I2ND Generic Enablers and the rest of the FI-WARE platform. The S3C GE is the central component, which interfaces all other three GEs (CDI, CE, and NetIC) to one common I2ND platform.

The interfaces provided externally (solid lines) by the I2ND Generic Enablers are used by:

- Applications: applications running on connected devices and interacting with the end user by means of proper graphic user interface. Applications make use of device features (i.e. sensors, profile information, stored data, etc.) as well as features or information accessible remotely (i.e. network services or cloud services) by means of the connectivity device capabilities;
- FI-WARE Services: business-to-business or business-to-consumer services provided by the FI-WARE platform that make use of the I2ND interfaces to get access to the underlying features, functionalities and information. In particular, the FI-WARE services are provided by other FI-WARE Generic Enablers that interact with the I2ND Generic Enablers;
- Cloud Services: they include both legacy cloud services as well as dedicated cloud services provided by the FI-WARE platform.

7.2 Generic Enablers

7.2.1 Connected Devices Interfacing (CDI)

7.2.1.1 *Target Usage*

The Connected Devices Interface (CDI) Generic Enabler (GE) will provide, to FI-WARE chapters and Use Case Projects applications and services, the means to detect and to optimally exploit capabilities and aspects about the status of devices, through the implementation of interfaces and application program interfaces (APIs) towards device features.

More specifically (see Figure 67) the CDI is a GE located in the connected device with the aim to provide to the network services and the developer's applications common APIs to exploit the device capabilities, resources, contextual information and contents.

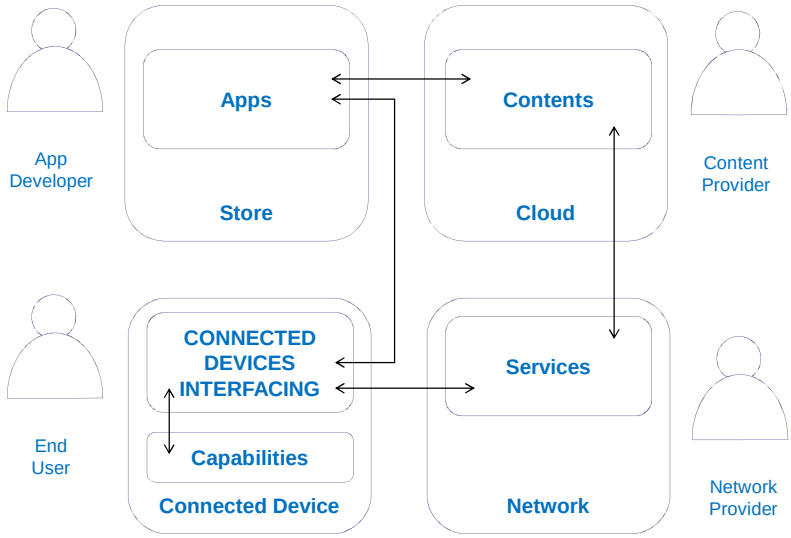


Figure 67 Connected Devices Interfacing (CDI) target usage

CDI will expose proper APIs to control all the capabilities and to get the related information of the connected device, including battery power, network status, location systems, quality of service, media capabilities, phone available features and sensors, profile information, status information and remote management facilities. As shown in Figure 67, as an example, the CDI can be exploited by the applications developers thanks to a common set of APIs exposed by the CDI allowing each app to get access to the device capabilities independently from the handset manufacturer, the OS vendor or the specific software embedded into the device. Also, the network operator can exploit the CDI APIs to remotely manage the parameters used by the device for establishing the connectivity to the network such as policies for access network discovery and selection, attachment and connectivity policies as well as management of the Quality of Service control and in-device routing policies in order to enable the mobile device to best match and react on its own to contextual situation (e.g. coexistence of WiFi and 3G coverage, usage of another connection over another access network for better QoS etc.) The result is that end user can enjoy his favourite contents, anywhere and anytime, using his preferred apps across heterogeneous networks and connected devices.

This means that same applications and network services can be used consistently over dissimilar connected devices, if they are equipped with the CDI. Moreover, the apps provided by FI-WARE application

developers will be able to exploit the capabilities and features the specific device platforms provide. This allows cloud hosted application services for example, to render their interfaces to make best use of individual devices and their relative connectivity. To realise this, the CDI GE will offer a uniform and standardized access and easy portability across multiple device classes of the features mentioned.

Here the term “connected devices” refers to a broad range of networked electronic components, including Handsets (cellular phones, smartphones), Tablets, Media phones, Smart TVs, Set-Top-Boxes, In-Vehicle Infotainment, Information kiosks, each being able to connect to a communication network.

To look at this in terms of practical benefits, a hypothetical scenario may involve a cloud-hosted media rich service which is accessed by many consumers across fixed or mobile terminal devices. The service provider will want to ensure that all consumers get the best possible experience, as a function of their connectivity, and the display and processing power (noting remaining battery power) of their device. By programmatically enabling the service with the ability to detect relevant details of the client device and its connectivity, decisions can be made at run-time in terms of selecting different levels of media ‘richness’ including sizing, resolution, 2d or 3d, etc., for individual users

Another example might relate to location-based services, where the location of the device (shopping mall), might be a useful indication of the customer’s intention, and suggest useful sidebar links in the form of advertising offers. Yet another might be the availability of triaxial accelerometers on the device and a user profile indicating a preference for this as a user input modality, which could be comprehended directly into the rendered UI configuration of a service.

7.2.1.2 *GE description*

Overview

The CDI GE is in charge of addressing a broad set of connected devices, not only the mobile ones, each adopting specific technology solutions in terms of hardware, software, middleware and runtime platforms, in particular concerning the development of applications.

A first challenge to be faced by the CDI GE is to provide homogeneous interfaces for application development. It is recognised that the extreme fragmentation of platforms adopted for connected devices, including a variety of different OSes and programming languages, is introducing several troubles to develop *once for all* the application and make it run on all such devices. As introduced in Section 3.1, different programming paradigms can be considered, which are exemplified in Figure 68. One step towards solution to fragmentation is represented by the adoption of middleware technologies (Java is one of the most relevant in this context), although not equally well supported by the majority of connected device platforms. Other emerging solutions rely on web based technologies available on most terminals. This trend seems to favour the development of applications which can run on web browsers or runtime engines.

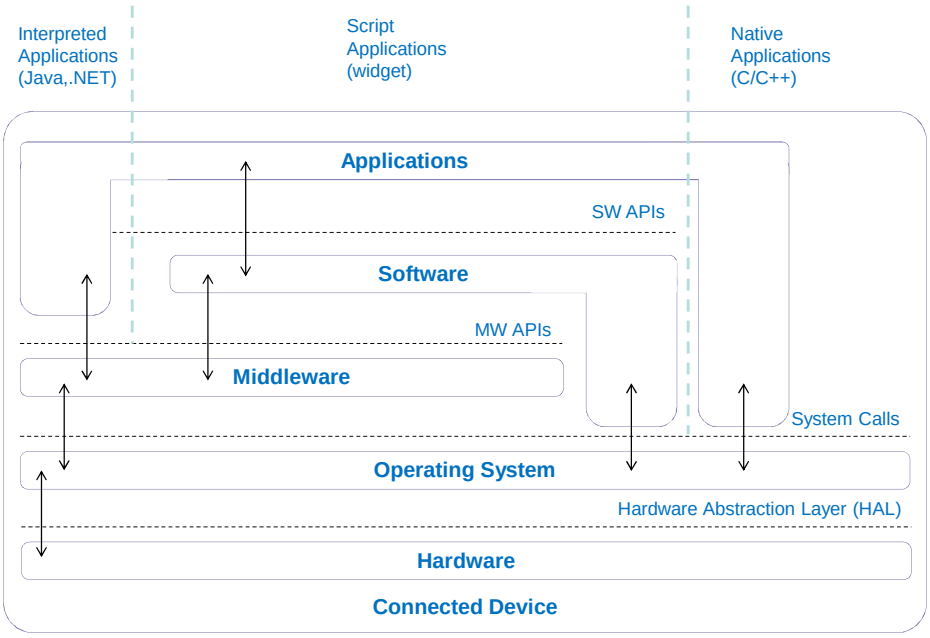


Figure 68 Applications programming paradigms

To cope with such heterogeneity of available but not winning solutions to fragmentation, the CDI GE tries to find a convergence point at least on the interfaces. The basic assumption for the definition of CDI interfaces is that they will be defined as much independent as possible from specific technology implementation and programming paradigm, thus creating a sort of layer on top of the technology dependent layer(s) of the devices that communicates with the applications and the network service by means of a interface layer as shown in Figure 69

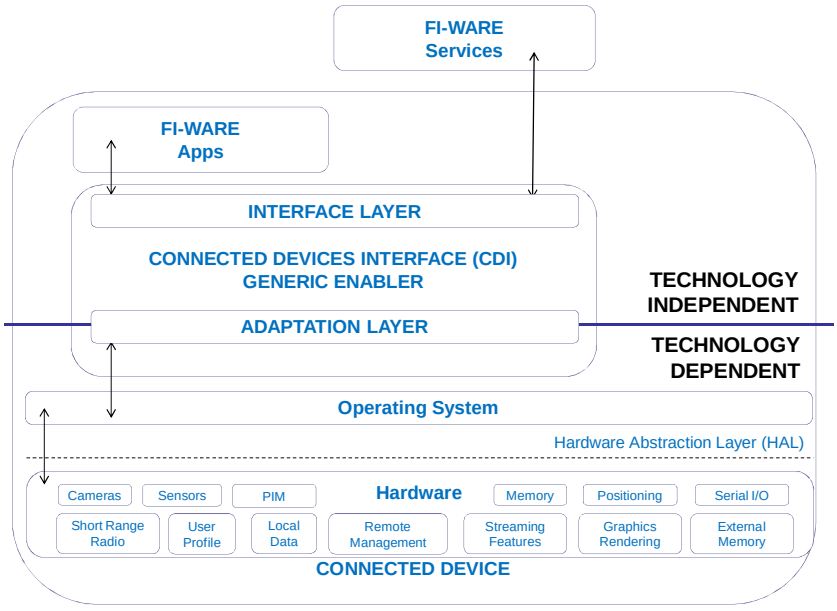


Figure 69 CDI virtualisation features of the connected device

As shown in the figure, the CDI GE interfaces the FI-WARE services and applications with the connected device capabilities, such as: embedded sensors (cameras, GPS, accelerometer, compass, etc.), connectivity (short range radio, radio interfaces, wifi, bluetooth, streaming, etc.), data (PIM, user profile, memory, etc.), management, graphics and so on. In order to cope with already available solutions, the CDI GE decouples a technology dependent adaptation layer from the technology independent interface layer. While the CDI GE interfaces are common for FI-WARE services and applications, their implementation and deployment on the connected device depends on the specific adaptation layer used to support the interface layer.

The adaptation layer can be implemented, as an example, as a stand-alone application (e.g. Java or C/C++), as a stand-alone run-time environment (e.g. OSGi), or as a web based run-time environment (e.g. WAC). Whatever solution is chosen for the adaptation layer it interfaces the connected device capabilities exploiting the primitives offered by the operating system. Thus the adaptation layer is technology dependent.

Instead, the interface layer relies exclusively on the required interfaces to support the FI-WARE chapters and Use Case Projects applications and services. The interface layer will not be defined from scratch. It will inherit the experience gained from other initiatives (such as WAC and W3C) and most of the interfaces will be re-used. However a gap analysis performed against the FI-WARE chapters and Use Case Projects requirements will lead to the identification of missing interfaces that will be formalised in the interface layer.

High level architecture

The high level architecture of the CDI GE derives from the convergence of current trends on widget execution environments architectures. An example of these standard, reference architectures is depicted below:

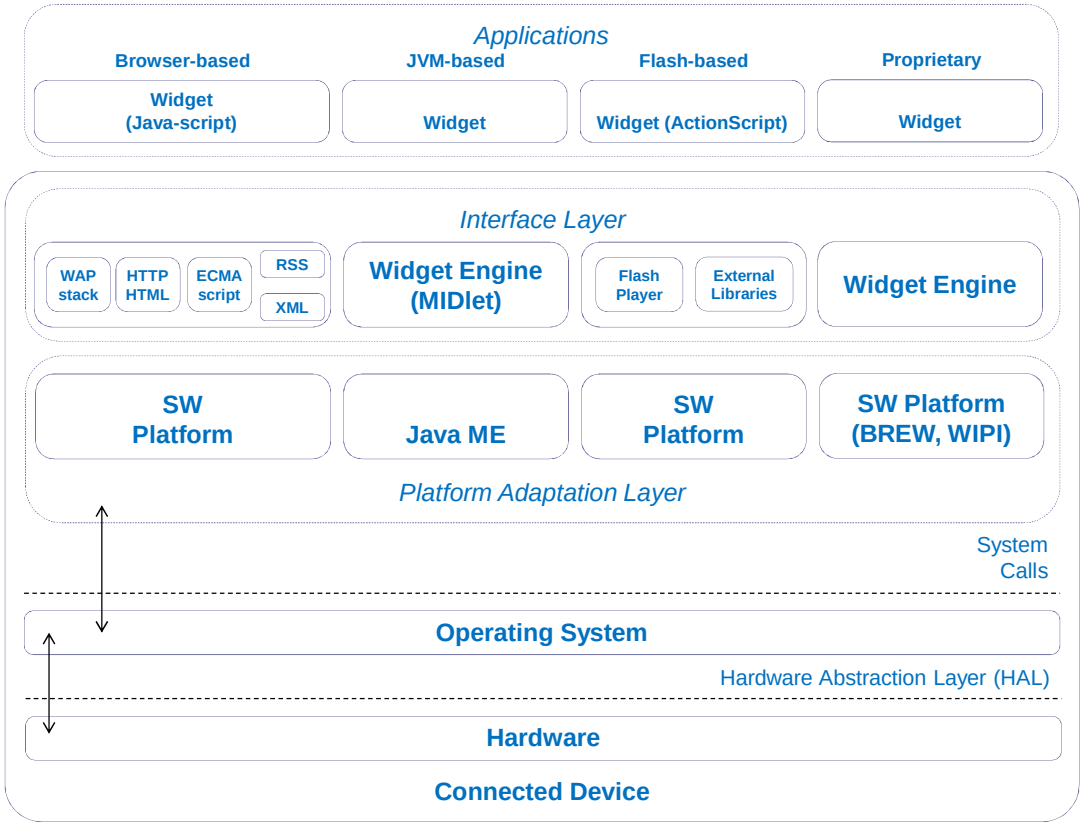


Figure 70 Reference execution environment for applications (widgets)

The architecture of the CDI GE, as anticipated earlier, is composed of two main elements: the interface layer and the adaptation layer (Figure 71). The first element, namely the Connected Device Interfacing Layer (CDIL), is in charge to provide to the higher layers the CDIL-NET and CDIL-APP interfaces. These interfaces can be further decomposed into a number of elementary interfaces, each exposing the specific APIs for a given feature of the connected device. The Platform Adaptation Layer (PAL) deals with interaction towards the specific platform, in order to adapt to the different architectures, programming paradigms, OSes, etc. This element can optionally expose the technology independent FI-WARE CDIL-PAL interfaces or directly interact with the Connected Device Interfacing Layer with proprietary, technology or language or OS dependent APIs.

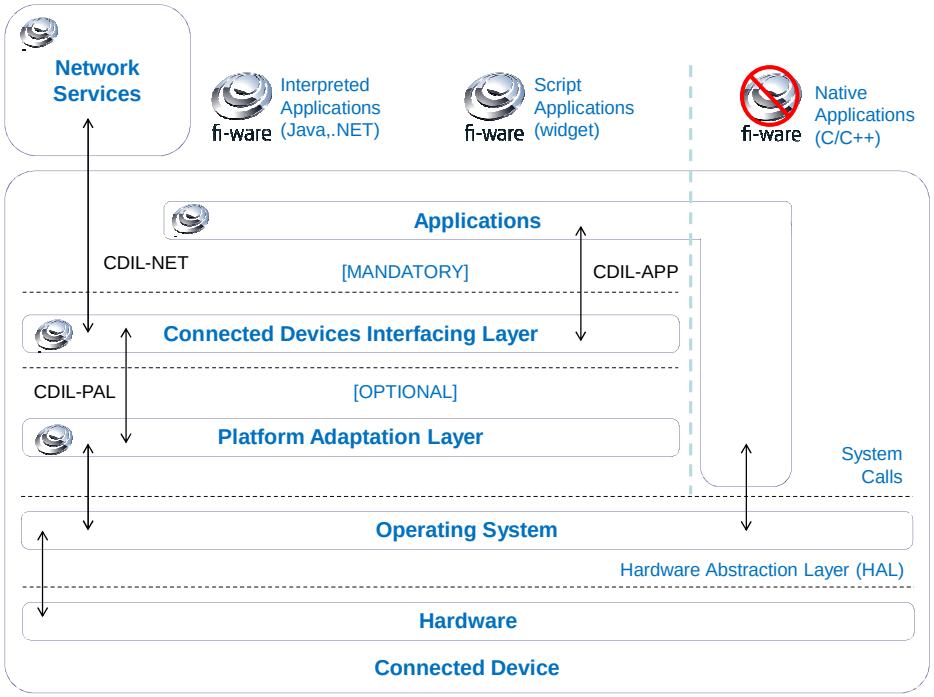


Figure 71 Layered architecture of CDI Generic Enabler

The **Connected Device Interface Layer (CDIL)** element exposes a set of interfaces providing open access to several features of connected devices. As shown in Figure 72 the CDIL exposes a set of interfaces to the apps created by the application developers (CDIL-APP), as well as to the network services owned by the network operators (CDIL-NET). On the other hand, the CDIL exposes a set of interfaces to interact with the underlying connected device capabilities, features, information, etc. (CDIL-PAL). It is worth to note that all the CDIL set of interfaces are totally technology independent. They do not rely on specific technology requirements such as hardware constraints, operating system limitations, programming language syntax and semantic or whatever. The CDIL interfaces are specified using a formal language (UML, IDL, etc.) and any implementation which is compliant with that specifications, can be part of the FI-WARE platform, execute FI-WARE apps and exploit in a transparent way the FI-WARE network services as well as the device capabilities.

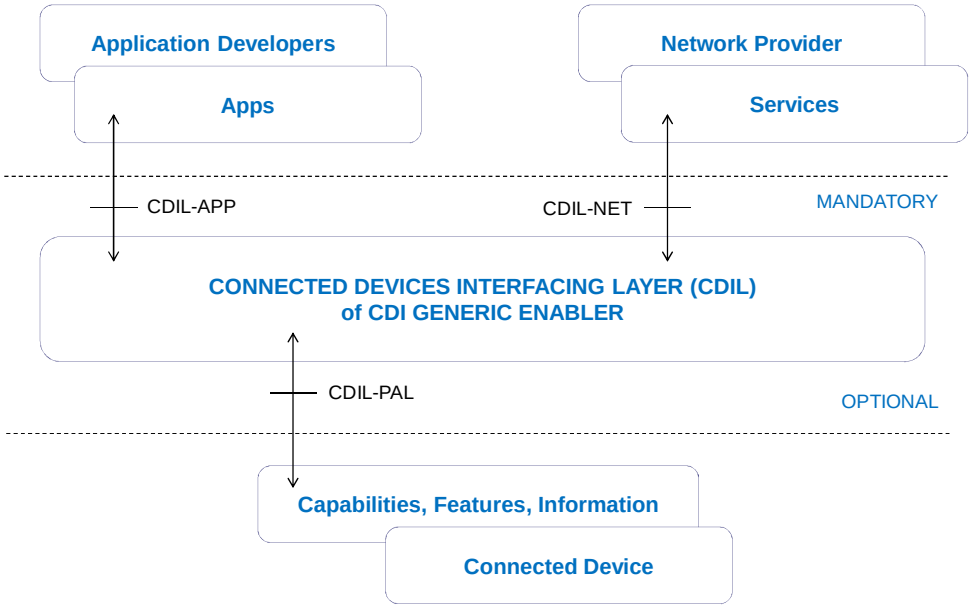


Figure 72 Overview of the CDI interfaces

In order to cope with the fragmentation of available technologies, networks, devices, sensors, applications, it is necessary to introduce well defined standardized interfaces among well-defined layers. This layer-isolation paradigm has assured some convergence especially in the telecommunication field. The TCP/IP represent at the moment a good point of convergence. But TCP/IP is only the basis on which applications, services, contents and information rely to be provided to the end users. End users enjoy their applications over dissimilar equipment, using heterogeneous operating systems, software and runtime platform. To limit this fragmentation, it is MANDATORY for the CDI GE to define proper CDIL-NET and CDIL-APP interfaces specifying their syntax, pre and post conditions, exceptions, semantics, implementation details, etc.. While the CDIL-NET and CDIL-APP interfaces are mandatory, is it not the same as for the CDIL-PAL. Indeed while it is mandatory to solve fragmentation at least at the top of the layered structure, just below the layer where application and network services run, it is not necessary to solve the fragmentation below. Thus CDIL aims to provide a mandatory set of interfaces usable by app developer and network operators to develop application and network services independently from the end user equipment. In order to provide the CDIL-NET and CDIL-APP interfaces the CDI must interact with the underlying hardware, software and middleware capabilities offered by the connected device.

The CDIL can optionally rely on proper CDIL-PAL interfaces in order to interact with the connected device capabilities by means of an intermediate **Platform Adaptation Layer** (PAL) that virtualizes the heterogeneous device hardware and software capabilities into homogeneous, syntactically and semantically specified interfaces. If the CDI adopts the CDIL-PAL interfaces is it fully technology independent. But a CDI implementation can also get direct access to the connected device capabilities, thus becoming technology dependent. It means that if a new hardware, software or middleware version on which the CDI is relying on is released, the whole CDI must be updated as well. Instead, in case the CDI adopts the CDIL-PAL interfaces, only the Platform Adaptation Layer must be updated, while the CDIL and its interfaces remains unaltered.

The FI-WARE decision to make the CDIL-NET and CDIL-APP interfaces MANDATORY has been done to solve the fragmentation problem, adopting an user-centric paradigm (where the user is intended the network service provider or the app developer). On the other hand, the decision to make the CDIL-PAL interfaces OPTIONAL, is to give the maximum freedom to the CDIL implementation and to guarantee at the same time the maximum backward compatibility with the already existent initiatives and standards (i.e. WAC) that do not rely necessary on a two layers (CDIL + PAL) architecture.

The architecture shown in Figure 73 is a functional view of what will be practically exposed to the users of the CDI GE, i.e. the FI-WARE network services and applications. The internal developments might be slightly different from the schematic view of Figure 73. Nonetheless, the main impact on the users will be hidden by the clear and open definition of the interfaces and related APIs on top of the CDI GE.

Clearly not all the interfaces to the device features are currently defined: the set of interfaces already identified in Figure 73 as interfaces which can enrich the application development of FI-WARE users comes from the identified needs by the project partners. However a deeper gap analysis of currently available interfaces and APIs against the required ones will be performed. What is found missing will be added in response to the requirements expressed by the Use Case Projects, and by the other sources of requirements the FI-WARE team is considering, and will refine the CDI GE architecture.

As a matter of fact, most device interfaces have been already considered by different standardisation initiatives (OMA, W3C, WAC, etc). The aim pursued in the definition of CDI GE is to operate in a sort of 'closed loop' taking as foundation of its architecture the wealth of information and specifications (even reference implementations) produced by such initiatives, adopting them wherever already well consolidated, encouraging their adoption through the FI-WARE Instances, extending those portions and features which are at the moment missing and, finally, promoting and supporting their inclusion in the standards evolution.. In this way, the CDI GE will be able to offer enhanced specifications to connected device features, that can be more complete and valuable to programmers, thus supporting successfully the wave of application developments of the Future Internet.

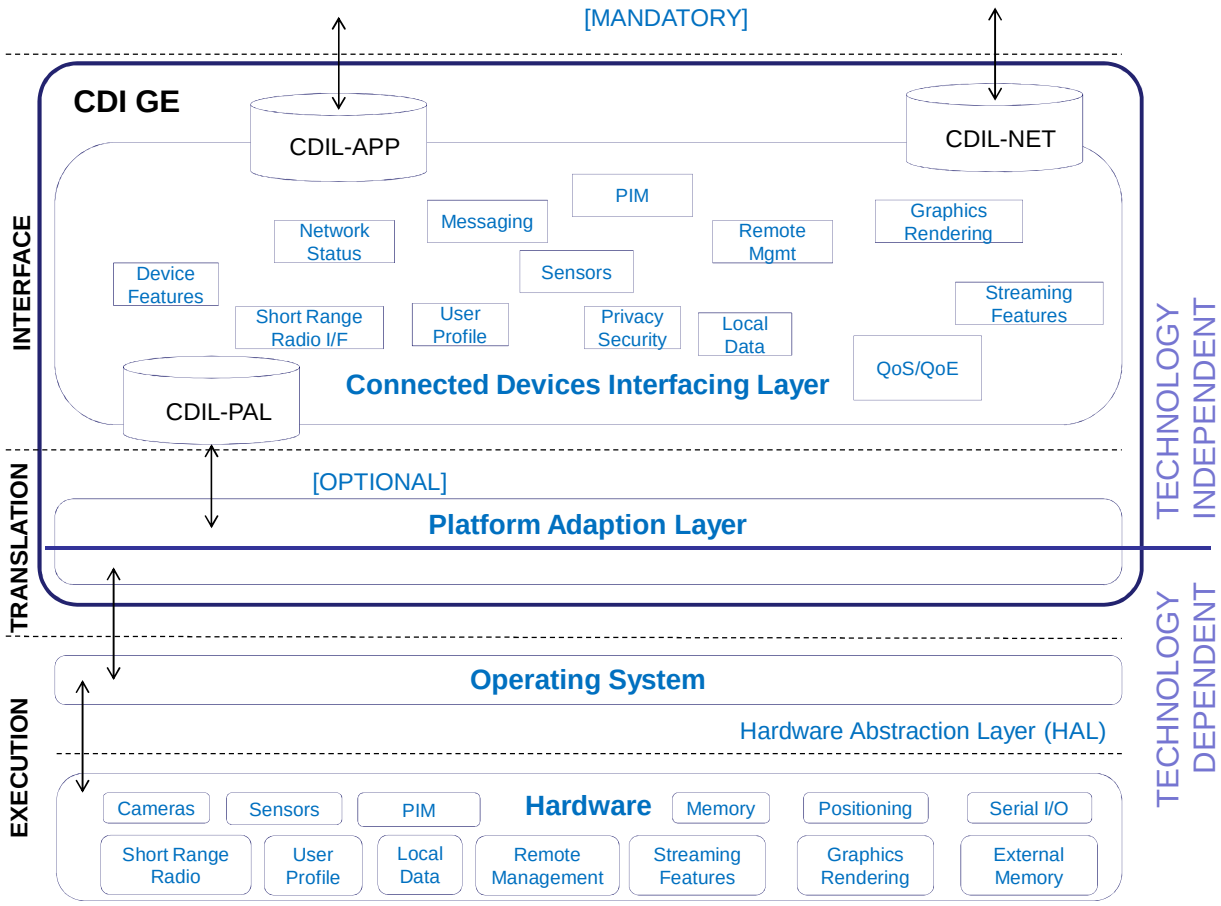


Figure 73 Functional architecture CDI Generic Enabler

List of functionalities

A number of functionalities will be provided by the CDI GE through the CDIL interfaces to the upper layers. A preliminary list includes:

- **Communication Services**
  - **Network Status:** of the device is represented by static information on the device itself (e.g. the number of device interfaces available), dynamic information on the connectivity (e.g. connected to 3G network) as well as the information received from the network operator on the possible available resources in the vicinity of the mobile device (e.g. a specific WiFi access in the vicinity can be used if more resources are required by the applications). This information has to be gathered from the network and from the device information and then presented in a unique form to the applications on the mobile device. Applications may need a connection towards the Internet, or towards other devices, in order to exchange information or to retrieve data or contents. Usually, devices are provided with several different communication interfaces (e.g., 3G, Wi-Fi, Bluetooth, etc.) and it is important to expose to the applications and services the communication features of the device, together with relevant information regarding the status of the connection and its availability.
  - **Quality of Service/Quality of Experience** (**QoE**) is a fundamental task for the IT Industry and it will play even more important role in the Future Internet. Quality of Service is an important feature for measure QoE but it is not the only one. The implementation of passive (like users' habits or zapping) and active (MOS, OneClick, etc.) estimations for measure QoE is a determinant step for refine it. Combining these elements will provide a better service for the end-users while the operators will be more reliable. A joint QoS/QoE approach must be faced off by the CDI GE, providing a feasible set of dedicated APIs to measure, at the end point, the QoS as well as the QoS perceived by the end user, thus the QoE.

- **Device Services**
  - **Sensors** are today part of the devices and are important as they provide a set of information that can be used by applications and services. As an example, the GPS receiver provides location information that can be used for location oriented application and services. Other examples of sensors usually integrated with today's devices are the accelerometer and the camera, but the interface will not limit the exposure of sensors, as it will be designed in a modular way so that it will be possible to expose also other sensors whenever available.
  - **Device Features (information about the device: battery, display, CPU, etc?)** FI-Ware and hosted application services will need to determine the features, capabilities and status of the connected device, in order to optimise the experience of the devices user. I/O status and options, screen resolution (and associated codecs), local computing and memory and remaining battery power, accessed via the CDIL API can be used to make run-time decisions on service rendering. Emerging features such as hardware enabled authentication or accelerated encryption can make end-to-end security less burdensome on the end user.
  - Inclusion of **Short Range Radio** (or proximity) technologies in the CDI interfaces eases the interaction of connected devices with IoT, enabling the interconnection and exchange of information with surrounding things and nodes. The radio capabilities should be able to manage virtually any kind of possible technologies. This is however a challenging aspect, and needs a thorough exploration during the interface definition phase, to include at least the most common technologies and those recently emerging (e.g. Bluetooth Low Energy, ZigBee).
  - **Privacy and security** aspects will be faced in conjunction with other chapters of FI-WARE (i.e. Security one), and will be in line with the approach followed by other initiatives (e.g. WAC includes mechanisms to securely manage how applications access device features, along with

supporting privacy policies as defined by W3C) and other standardization bodies. This interface will thus deal with the secure access to device features and user's data, as well as to assure the origin and the integrity of the application and service running on the device.

- **Remote Management** aspects will be considered as to empower the mobile device with information on the vicinity network status including access networks discovery information, inter-system mobility and routing policies enabling the selection and the forwarding of data traffic from the mobile device to the most appropriate access network based on the connectivity requirements of the various applications on the mobile device and according to the indications of the network providers.

- **Personal/Data Services**
  - **User Profile (Identity, Authentication, etc)** information is fundamental to provide advanced user-aware services, it is hence necessary to correctly and safely develop a functionality to identify the user of the device. This is a functionality which requires a strong cooperation with the general security aspects provided by the FI-WARE developments (see Security chapter).
  - **Access to Personal Information Management (PIM)** provides a functionality to access, add, remove or update information there contained. Elements of PIM include the **Calendar** (collection of events described in specific formats, according to standards definitions like e.g. RFC 5545) **, Contacts** (information about a person, including e.g. phone numbers, email addresses, etc) collected into an Address Book and **Tasks** (a list of items to be done/completed, each described e.g. with a priority, deadline, etc).
  - Access to **Local Data** of a device will be ensured by this interfacing functionality. This will enable the application developers to get access to and manage **e.g. pictures, videos, data files, applications**, etc stored on the device. One particular aspect to be carefully evaluated is to limit the access strictly to specified data, making sure no other portions of local data will be affected in any way.
  - **Messaging capabilities** like sending messages through different technologies (SMS, MMS, Email, etc) are among the most typical device capabilities which can be exploited by an application, therefore corresponding APIs are necessary to satisfy such requirements.

- **Media Services**
  - **Graphics rendering (3D, HD, etc))** rich media and graphically oriented application services will need to be both interoperable (through scaling) with a broad range of devices, but also to make the best use of facilities on the more advanced devices. Determining the capabilities of different device classes at run-time enables the most optimal rendering for the end-user.
  - **Streaming Features (transcoding capabilities, etc)** similar to the Graphics Rendering situation, the presence of onboard accelerators or specific codecs can signal to a hosted service that a particular device class is capable of receiving a richer format of graphical or other data. Such a decision would need to comprehend availability and cost of the necessary bandwidth also.

7.2.1.3 *Critical product attributes*

- Consistent access to device capabilities and context from both hosted and native running services regardless of device class or middleware/OS, through a simple API layer.
- API layer to be readily extensible to accommodate future device classes and software stacks, resulting in significant efficiency, scalability and flexibility benefits for service providers and application developers, and expanded choice and optimal Quality of Experience for service consumers

7.2.2 Cloud Edge

7.2.2.1 *Target Usage*

The concept of Cloud Proxy has been introduced in the "Cloud Hosting" Chapter. We just remind here the concept of cloud proxy and a few associated illustrative use cases.

The concept of cloud proxy comes from the assessment that, despite the apparent inexhaustible computing or storage resources offered by the cloud concept, the link between the end-user and the cloud remains unique, providing a relative low bandwidth (compared to the internal home network bandwidth, in particular). The idea consists in using an intermediate entity, called cloud proxy, located in the home network and provided with computing and storage capabilities. The cloud proxy can therefore take benefit of the high speed connection with each device of the home network and is in a position to act as a Cloud agent closer to end devices.

To illustrate this concept, we also described some use cases in the Cloud Hosting chapter. One, from the cloud to the user, consists in pushing part of a VOD catalogue's content in the cloud proxy (including movies trailers for instance). The purpose is to accelerate the browsing of the database which may be considered as a key element of the VOD application attractiveness.

Another use case, from user to cloud this time, consists in, when uploading data into the cloud, to first upload it in the intermediate storage offered by the cloud proxy. Time required to upload data directly in the cloud may be relatively long, while relatively short in the cloud proxy (hours versus minutes for the same amount of data). Using the intermediate storage of the cloud proxy makes possible for the user to leave the home network after the few minutes required for the local upload, while the cloud proxy shall then be in charge to upload data in the cloud as a background task.

Figure 74 illustrates the concept of cloud proxy.

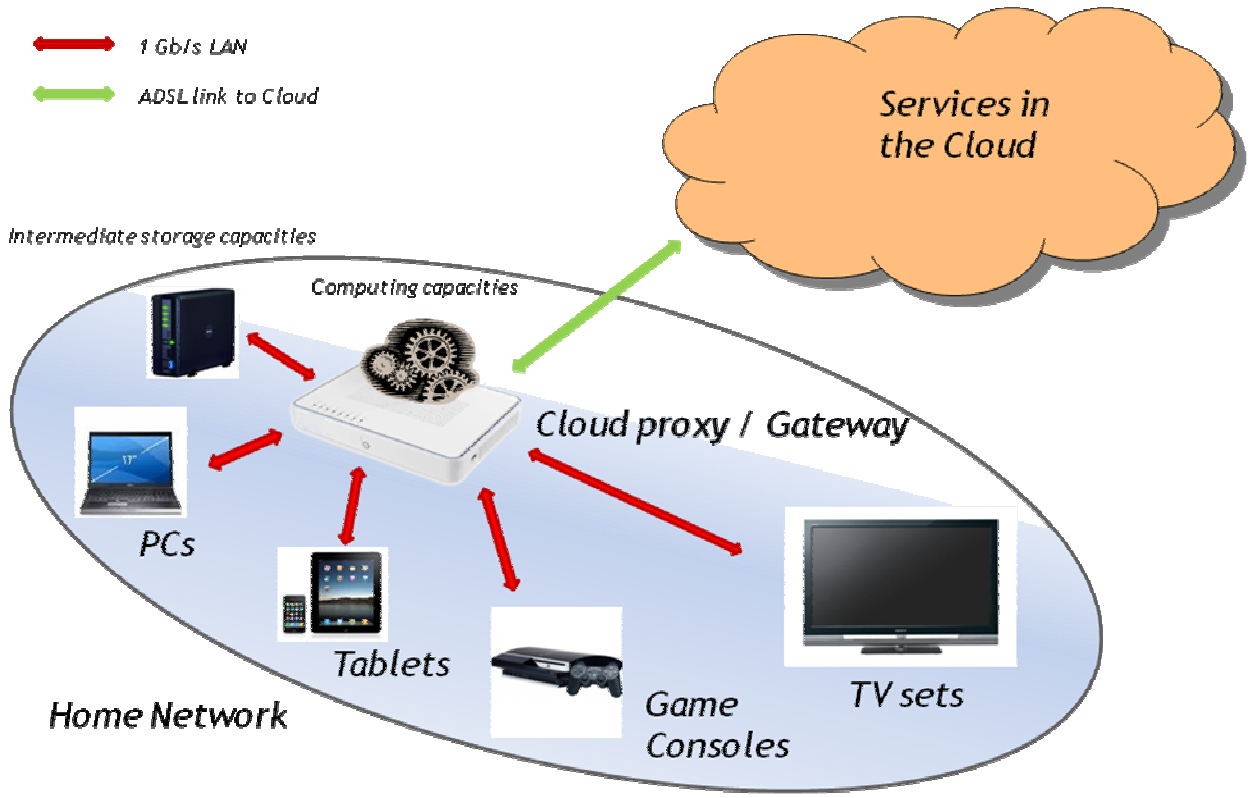


Figure 74 Cloud Proxy concept

The IaaS Cloud-edge Resource Management GE defined in the Cloud Hosting chapter comprises those functions that enable to allocate virtual computing, storage and communication resources in cloud proxies. However, FI-WARE will not only define and develop the software linked to the IaaS Cloud-edge Resource Management GE but also software linked to middleware technologies and common facility libraries that will be used in VM images to be deployed in cloud proxies. These middleware technologies and common facility libraries are defined and developed with the I2ND chapter and described in the following section in more detail.

7.2.2.2 *GE description*

We give here a high level description of the different functional modules and associated interfaces of the cloud proxy, as represented in Figure 75

End-device communication

End-devices do implement various communication protocols. Consumer Electronic devices may preferably use UPNP/DLNA protocols, while PCs may use NFS or SMB-CIFS depending on their Operating System (Linux/Windows). Lastly some devices like iPhone may require a dedicate software for communication as far as they do not allow a direct communication with the file system level.

The end-device communication module is in charge of implementing ad-hoc protocols and software in order to ensure an optimized data exchange between end-devices and the cloud proxy. Exact list of used protocols has to be precised.

Home system management

UPNP is a zero-configuration protocol, requiring no user manipulation to establish the connection between two devices. This is not the case for SMB or NFS and many other protocols which the end-device abstraction layer may implement. In order to avoid to the user complex manipulations requiring specific skills, user manipulation required by the various communication protocols shall be automated. This is the first task of the Home system management module.

This module shall also be in charge of creating ad-hoc storage spaces on the end-devices as well as on the cloud proxy storage capabilities, and possibly perform various monitoring task on these devices. It shall eventually report to the applications the resources available in the home environment.

The home network management module will be in contact with other modules of the cloud proxy, in particular the Outside communication module and the end-device communication module. APIs between these modules shall be defined. Exact content of these APIs have to be detailed.

Outside communication module

We just mentioned that the Home system organisation module may report information to applications regarding resources available in the home environment. Exchanges with the Cloud may also concern end-devices or the permanent storage module via the end-device communication module, when a cloud application may download data in the home system for instance.

Outside communication may also concern inter-home system communication. In case of distributed application, cloud proxies may directly interact together.

A set of APIs shall be defined, comprising API to the home system organisation module, basically for monitoring or resource description, and API to end-devices and permanent storage, basically for data exchange. The APIs may include the Connected Device Interfacing Layer (CDIL) as defined for the CDI GE. Exact content of theses APIs has to be precised.

Virtual machines management

Cloud application developers may wish to download and execute piece of software in the cloud proxy. For the sake of security (both in term of insulation of data and software execution) and flexibility (installation of a service close to the user), the cloud proxy may support the deployment of Virtual Machines (VMs). The detailed definition of this module can be found in, the FI-WARE chapter devoted to Cloud Hosting. We however mention it here, in order to provide a complete picture of components running on the cloud proxy.

Protocol adaptation

The “outside communication”, “end device communication” and “device management” blocks include a “protocol adaptation” subfunction. This subfunction can translate various protocols to an unified internal vision that is much easy to manipulate, it can also give the cloud proxy the opportunity to embed protocols inside protocols (for example, do some tunnelling). It must be also noticed that in some cases, this “protocol adaptation” can act transparently, even for non-standard protocols. For example, a virtual machine deployed on the cloud proxy could implement a iTunes-compatible module and could transparently communicate with an end-device such as an Apple product (iPad, iPhone …). Another example is the implementation of the “Internet of things” concepts inside a virtual machine that could include protocol / language interfaces to non-standard home-automation devices.

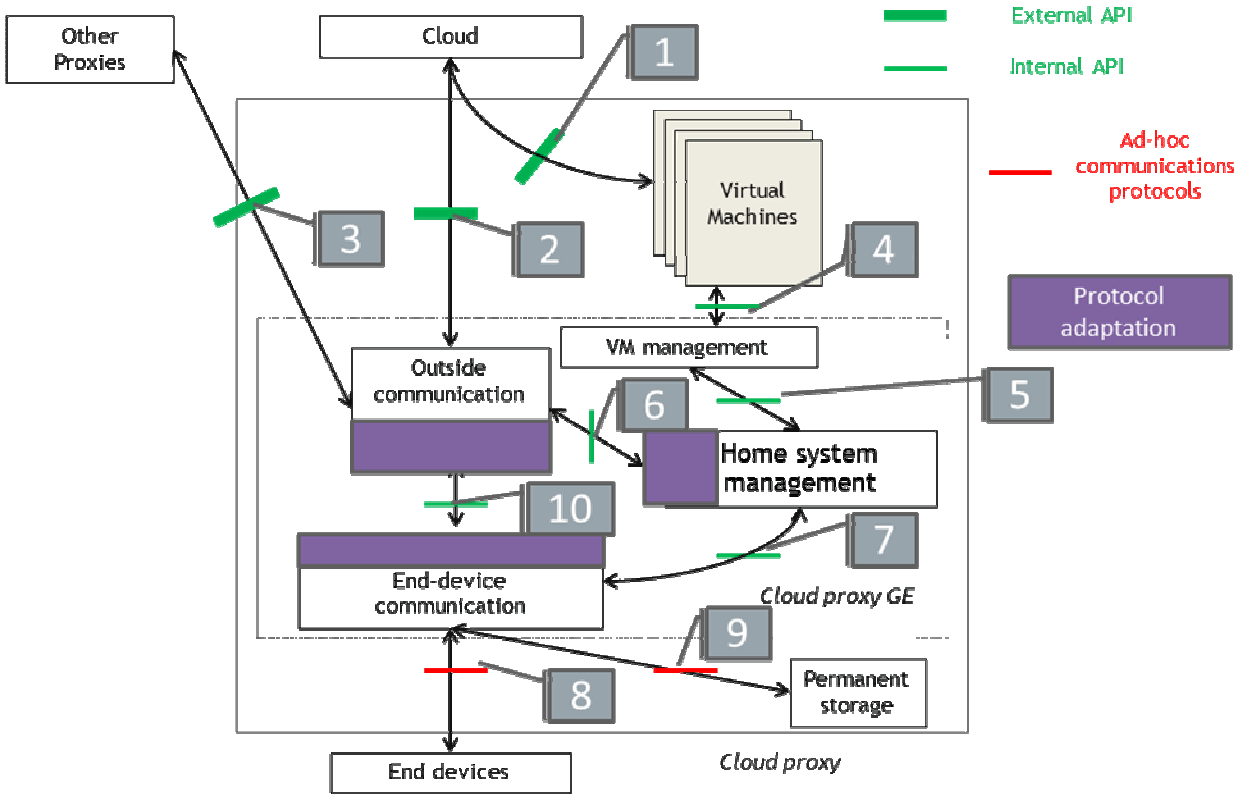


Figure 75 Cloud proxy main functional modules and APIs

Summary of Interfaces

The set of interfaces related to the CE GE is identified by the numbers in Figure 75. We can broadly divide the interfaces in internal (i.e. inside the Cloud Proxy GE and not exposed to outside entities) and external (i.e accessible by outside entities). Internal interfaces can further be related to Cloud Hosting chapter implementation or shared between that chapter and I2ND. The former are listed here for the sake of uniformity, but will be taken care of in Cloud Hosting chapter, the latter will be part of the CE GE. Here is a short description of the functionality they implement:

1. (external): interface between the applications hosted in the cloud proxy and external applications → these interfaces depend on the application and cannot be standardized. They are application-dependent, for example, a specific application can be architectured in such a way it is executed partly in the cloud and partly inside the cloud proxy, the interface between these 2 parts depends on the application architecture itself. The cloud proxy provides the basic communication protocols to implement this API
2. (external): interface between external applications and the cloud proxy → all the communications between external applications and the cloud proxy itself (except for the VM-related and specific communications) are transported through this interface. This API will provide ways for the cloud-based application to communicate with the management features of the cloud proxy and will support transparent or adapted communications to the end devices
3. (internal): communication with other cloud proxies → a potential extension allowing applications running on different proxies to communicate using optimized protocols and also enabling multiple cloud proxies mutualising their resources in order to support Cloud Hosting functions.

4. (internal): interface between the VMs and the VM-management entity → this API allows the cloud-based application and/or the cloud proxy itself to interact and to manage the VMs (life cycle management (load, unload, launch, kill and get/set status of the VMs).
5. (internal): interface between the cloud proxy management and the VM management entities → transport of the life cycle management controls to/from the VMs
6. (internal): interface between the outside communication entity and the management → transports the various management / control / status coming from externa applications or the Cloud Hosting functions linked to VM management.
7. (internal): interface between the inside communication entity and the management → same as 6. Allows local devices to have interaction with the management entity
8. (internal): communication with the local devices → this interface is device-dependant (ie: an iPod will not interact the same way a uPnP/Dlna device will). Support of standard protocols will be the 1st priority, especially uPnP/Dlna
9. (internal): interface to the local storage → this interface will give access to permanent local storage. Available APIs (VFS, NFS, FTP …) will be used
10. (internal): interface between outside and inside worlds → this interface will transport data and controls between external applications and the local devices (main channel). The protocol adaptations will allow protocol / language transformations as well as possible tunnelling if required.

7.2.2.3 *Critical product attributes*

- Cloud proxy as evolution of the home hub, able to federate the connected devices and expose functionalities to support a large variety of service bundles
- API layer provided by the Cloud proxy to allow the cloud-based applications to interface properly and easily with devices associated to the home environment and manage/access to local data.

7.2.3 Network Information and Control (NetIC)

7.2.3.1 *Target usage*

The Network Information and Control (NetIC) Generic Enabler will provide to FI-WARE chapters as well as usage area applications and services the means to optimally exploit the network capabilities, by means of the implementation of an interfaces and APIs towards networking elements. NetIC will both expose related network state information to the user of the interface as well as offer a defined level of control and management of the network.

The beneficiaries of the interfaces include, among others, content providers, cloud hosting providers, context providers/brokers, and (virtual) network providers/operators, which need to have information about the network between them and their clients and which might want to set up flows/virtual networks between them and their clients and may want to control such flows/virtual networks in order to respect pre-defined Service Level Agreements (SLAs), for example in terms of provided Quality of Service (QoS). There are several use cases for the NetIC Generic Enabler, for example the following:

- A cloud hosting provider has a couple of data center locations. In order to distribute the allocation of VMs and applications to the various locations, the cloud hosting provider should know about the characteristics of the paths between the locations (delay, available capacity). To get this information, the cloud hosting provider can request via NetIC from the network provider (regularly or per

scheduling event) the characteristics of the paths between his data centers. NetIC will provide the requested information by a corresponding interface. In addition, when dealing with migration of VMs and applications across data centers, the cloud hosting provider may request to setup temporal virtual private connections with a certain quality of service being guaranteed during the time of migration.

- To deliver a service to a client, a service provider may need a certain minimum link quality, e. g., for a high-definition live video streaming service. If the client is willing to pay for this, the service provider will request via NetIC from the network provider the setup of a virtual connection with certain quality characteristics between the server and the client. NetIC will do so if capacity is available.
- A network service provider wants to implement new business models based on the "pay-as-you-go" paradigm, setting up a specialized service for a group of clients. The specialized service is built orchestrating the network resources dynamically. For this he needs a virtual network (optical or packet based) connecting some servers, network elements and the involved clients, potentially running customized protocols. The service provider can request via NetIC from the network provider to setup a virtual network between the involved endpoints, possibly also with some specified constraints (quality characteristics, isolation against other virtual networks, energy efficiency metrics)
- A service provider wants to set up a specialized service for a group of clients. For this he needs a virtual network connecting some servers and the involved clients, potentially running customized protocols. The service provider can request via NetIC from the network provider to setup a virtual network between the involved endpoints, possibly also with some specified quality characteristics and isolation against other virtual networks.
- A cellular service provider wants to run its business on top of a virtual network which is able to “breathe” (to be re-configured as demand changes) since load in idle and busy hours differ significantly. Benefits are reduced expenses (CAPEX turned into pay-per-use OPEX), reduction of energy consumption and management flexibility. Today mobile traffic is mapped into static MPLS tunnels, and the infrastructure providing these tunnels are owned by the cellular service provider, too.

A fundamental challenge for the implementation of NetIC is that the network functionality is typically distributed over different elements potentially implemented internally in different ways (in multi-vendor environments), and that the interfaces have to take into account the constraints of different providers (in multi-network service scenarios) as well as legal and regulatory requirements. This problem has been solved in the past by different standardized control plane solutions. This readily available functionality could be re-used by NetIC in order to provide a smooth evolution path rather than a disruptive revolution. NetIC instances may be deployed by the different involved parties (e.g. virtual network providers/creators, and virtual network operators/users running a business on top). As a consequence, several instances of NetIC with different scopes may have to work together to deal with a request from e.g. a service provider or an application. Each might cover a different part of the network, for instance in horizontal direction (i.e., type of access) or in vertical direction (i.e., ownership, virtual network).

7.2.3.2 *GE description*

NetIC will implement interfaces (potentially split into several sub interfaces in later phases of definition) that provide open access to network status and forwarding functions inside the network (see Figure 76). NetIC will provide an abstraction of the physical network resources, as well as the capability to control them (up to certain limitations given by the physical resources’ owner and operator), opening the way to the utilization of the same network infrastructure by different (even virtual) network providers. Within these limitations, a key advantage of open networking is the possibility for each different network operator to implement its tailored network mechanisms even relying on the same common network infrastructure (own addressing schemes or protocols, dynamic tunnels, etc.). Also, premium network services (in respect to virtual network providers) can be implemented, e. g., for the interconnection of private and public clouds by dedicated links with guaranteed Quality of Service (QoS). The NetIC GE provides an interface for dynamically controlling network QoS parameters inside one administrative domain. The interdomain provisioning of QoS guarantees will likely have to use the API exposed by the S3C GE, since it requires authentication, authorization, accounting, and SLA conformance checks.

On the one hand, NetIC will provide access to network status information. Network status information will be used by the users of the interface in order to both collect statistics regarding the utilization of the network and to acquire near real-time information about the status of the network. As for example, typical network status information of interest for the users may be related to the network topology, interface status, and end-to-end path performance in terms of delay, jitter, packet loss, available bandwidth, etc.

The capability to acquire information from the network will enable service providers or applications to either scale their services to the effective conditions in the network to deliver optimum Quality of Experience (QoE) or to select the best suited access point for a certain service. In particular, resource management algorithms as well as other optimization algorithms (e.g. algorithms related to mobility management and handover optimization) will benefit from near real-time input information describing the status of the network. The information will be consolidated in the abstraction layer, where the network operator will bundle the information to forward the respective parameters to the usage area specific Enabler platforms and the different core-platform entities.

On the other hand, NetIC will provide access to the forwarding functions of the network nodes. The advantage of having this is twofold. As an immediate result, the capability offered by NetIC to have a direct and controlled access to nodes' forwarding functions will enable a certain level of programmability within the network, fostering the Software Defined Networking (SDN) paradigm. According to the SDN paradigm, the user of the interface can develop software application to automatically control and manage the network on the basis of its rules and policies, e.g. flow processing, routing, and addressing. Furthermore, there can be access to the network resource management functions. At the same time, NetIC will also provide the mean to enable network virtualisation to open a controlled way for virtual network provider operations. Figure 76 shows the main NetIC functions and APIs of NetIC.

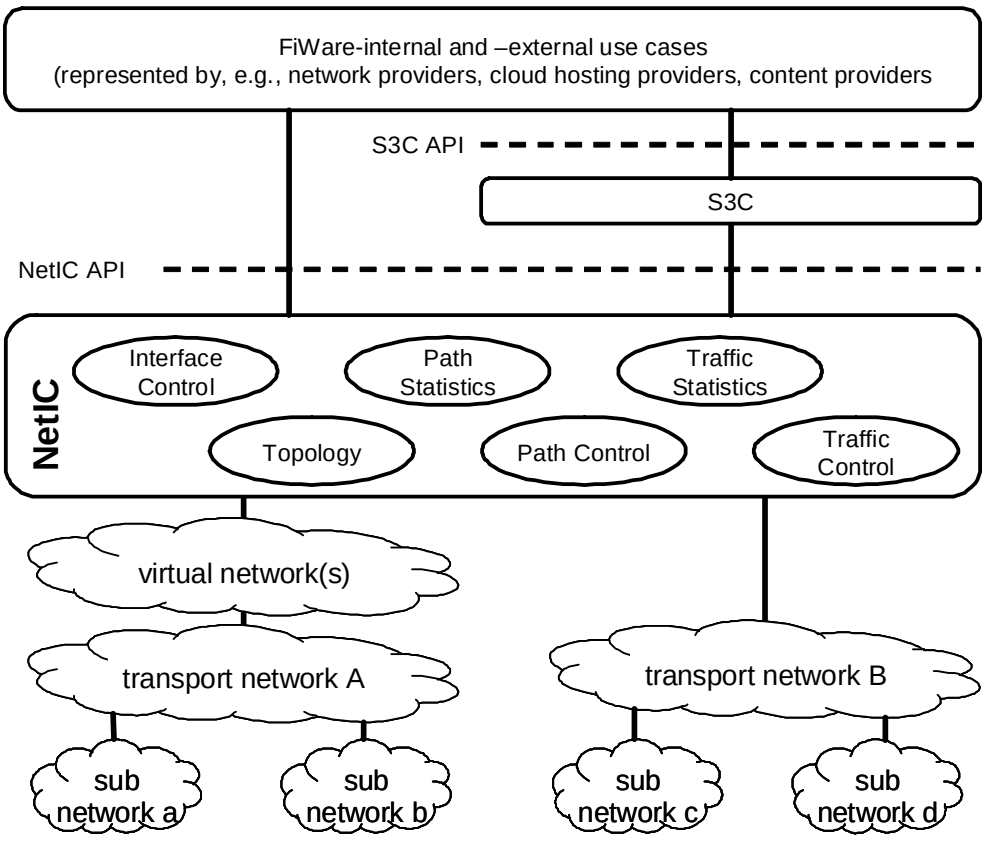


Figure 76 Main NetIC functions and APIs

The NetIC Generic Enabler is expected to provide access to the following functions:

- **Interface Control**: This function will provide information about the status of a network element interface. Callers may also be able to change selected parameters. An example would be interface activation or deactivation.
- **Topology**: This function will provide abstract information about the nodes, edges, and how they are interconnected. Furthermore it may allow the modification of network topology. An example for the latter case would be the setup of a virtual network.
- **Path Statistics**: This function will allow querying the properties of an end-to-end path. An example query function would be an estimation of the available bandwidth on a path, or its packet end-to-end delay.
- **Path Control**: This function will provide mechanisms to change the current status of a path. The realization of this function as well as its actual scope of operation will be technology dependent due to, e.g., the different realization of packet-oriented and circuit-switched networks. It will affect the setup, modification, and teardown of paths. An example control function would be the definition of a customized routing scheme via the OpenFlow interface. This supports implementation of QoS by isolation of paths (IntServ).
- **Traffic Statistics**: Various types of information about the network usage may be of interest. This function will provide access to such statistics and potentially a control over the monitoring process.
- **Traffic Control**: This function will allow influencing, e.g., the handling of different traffic types in the network (e.g. priorization of traffic according to DiffServ). This enables differentiated provisioning of QoS.

There are further relevant network functions that might be of interest to be exposed via NetIC API, and further analysis is required whether corresponding interfaces will be needed. For instance, address resolution could be a potential use case to find out whether a given address is present in a network.

Depending of the network, the types of nodes, and policies, only a subset of these functions may be available to a user of the NetIC interface. Therefore, the NetIC generic enabler will also provide a mechanism that will advertise the actually supported features.

The challenge for providing network status information is to find a good trade-off between accuracy, timeliness, and overhead. Providing accurate information and a global insight of the network situation may require extensive continuous measurements. Furthermore, data may be unreliable, incomplete, inconsistent, or misleading, or not even be available due to operator policies (inter-domain topology hiding paradigm). In this respect, NetIC has to address several open issues in order to provide to its users the information they really may need with an adequate level of accuracy. Several techniques to obtain network measurements and status information will be considered, spanning from active techniques to passive approaches.

The monitoring process regarding physical resources consists of gathering metrics about the state of the resources available on each network element that is part of the monitored infrastructure. This is performed by a monitoring agent installed on each element, which will send the gathered data to a collector for processing and persistence.

The challenge for providing some means for network control is that such a generic interface should be technology and implementation independent as far as possible. Specifically, it should support packet switching as well as circuit switching, e. g., in optical networks. This is challenging since networking technologies differ in the granularity of capacity that can be allocated. There are existing and emerging standards in that domain, such as General Switch Management Protocol (GSMP) or the Forwarding and Control Element Separation (ForCES) standardized in the Internet Engineering Task Force (IETF). Another interface is OpenFlow, which is driven by the Open Networking Forum (ONF). But all those solutions are still limited to specific networking environments and they offer limited control and management capabilities. They also have limitations concerning scalability and flexibility, and they are still not well integrated into cloud management solutions. In this respect, NetIC aim is to define a proper interface to enable network control, which will be based on the mentioned available protocols and will maybe extend

them in order to cope with all the requirements coming both from other FI-WARE GEs and applications running on top of FI-WARE Instances.

The challenge of common sets of parameters well understood by the different usage areas will be the translation. The underlying network infrastructure will follow different standardisation rules. Currently the Internet world follows mainly the IETF, the mobile communication follows 3GPP, whereas the fixed line world have standards from ETSI, ITU-T and IEEE. All of them have to be translated into a common language and interface description.

The NetIC generic enabler targets open networks that expose interfaces. Technological limitations, network operator policies, or other constraints may prevent such open interfaces. For instance, the NetIC generic enabler may not be applicable in walled garden environments, such as 3GPP cellular access networks. In that case, certain functions may partly be available by other means, e. g., by the S3C generic enabler. In general, the NetIC and S3C generic enablers will have to be aligned. Most notably, the inter-domain usage of NetIC functions may require authentication, authorization, and accounting, for instance by using the corresponding service-level interface of the S3C generic enabler, which then calls the NetIC interface.

7.2.3.3 *Critical product attributes*

- The NetIC Generic Enabler will provide to its users access to network status information. Interfaces available today are already able to provide specific information, but the interface highly depends on the specific network technology. The aim of NetIC is to define a set of general functions to access network status information in a technology independent way, overcoming the heterogeneity of today's solutions.
- There are several standard technology and implementation dependent interfaces to control and manage specific networks. To overcome this heterogeneity, the objective of the NetIC Generic Enabler is to provide a generic interface to control and manage open networks, which shall be technology and implementation independent.

7.2.4 Service, Capability, Connectivity, and Control (S3C)

Target usage

The Service, Capability, Connectivity, and Control (S3C) Generic Enabler is the manifestation of an adaption layer between the targeted network control layer for Fixed-Mobile-Convergence: Evolved Packet Core (EPC) and all possible applications and services.

Driven by the roll-out of new wireless access technologies providing only packet transport capabilities like the 3GPP Long Term Evolution (LTE), the future massive wireless broadband environment is bound to transform into a data dominant environment. In order to respond to the requirements of the new environment, 3rd Generation Partnership Project defined the Evolved Packet Core (EPC) as a new IP connectivity control platform enabling wireless access network diversity (including LTE, UMTS, WiMAX, WiFi etc.) and offering seamless connectivity for the various service platforms. It maintains the same central concepts as previous 3GPP architectures, like IMS: policy based QoS and charging, subscription based access control, handover support and security, offering a scalable alternative to the current deployed architectures.

By using all-IP based communication protocols and functionality, the EPC is designed to support large number of subscribed devices, their signaling and data exchanges through its flat network design supporting mobility management inside the same or in different access technologies, subscription based resource management and accounting along with security support of the communication.

EPC provides a transparent convergent network layer for the IP Services. From the perspective of a service provider without a modification of the way the services communicate, it enables a high degree of satisfaction, by transparently supporting features like access control, QoS assurance, seamless mobility between the different access networks, prioritization and security.

Also due to the resource reservation mechanisms, the services have a guaranteed quality of the communication which is an addition to the typical IP communication and a high added value for broadband communication on mobile devices with reduced processing power.

EPC provides also a set of control mechanisms between the service platform and the network core. Through these mechanisms, the EPC aware applications can transmit indications on the resources that have to be reserved for the specific users at specific moments of time. They can also receive upon request information on events happening at the link and network layers e.g. the connected device lost connectivity or a handover to another access network occurred. By these mechanisms, the applications can be adapted to the momentary context of the mobile device and to offer services customized not only based on the service level user profile, but also to the mobile device in use, the mobility pattern and to the surrounding network context.

Although not yet standardized, EPC is able to export a set of enablers to the applications which offer even more flexibility in the service delivered to the mobile user. For example, services may use the location of the connected device or even ambient information on the vicinity of the connected device and the subscriber identity of the mobile device, in order to further more adapt to the environment conditions and to ensure a more secure communication.

With the deployment of new devices such as sensors and actuators along with the further increase in usage of the IP capable mobile devices such as laptops, tablets and smartphones, it is foreseen a high increase in signaling and data traffic expanding the overall required resources from the network.

Also new service paradigms are expected to be further developed such as mobile cloud computing or machine-to-machine communication which will even more broaden the types of communication both as communication patterns, mobility and resources required.

However, the EPC was designed with point-to-point human communication as the main service paradigm which will require a new adjustment in customizing the IP connectivity according to the momentary needs of a high number of devices using the broad future internet services.

7.2.4.1 *GE description*

The S3C Generic Enabler comes to mitigate these challenges of the packet core networks by offering an extended API for the various service platforms and by dynamically adapting these requirements for the packet core network. This adaptation includes novel optimized data transmission mechanisms for broadcast/multicast and for small data exchanges as well as mechanisms for data caching and aggregation. S3C also provides the control of the network resources through the inclusion and the orchestration of the new management functionality addressing both the subscribers and the services such as network identity and network connectivity, events, resources, charging and inter-carrier management, security AAA and accounting as well as network context data management such as location enablers.

Currently, the EPC has an open API through the Diameter protocol enabling a minimal synchronous reservation of resources which implies that any communication has to be signalled between the mobile device and the service platform as well as between the service platform and the core network control. However, for legacy services, such an interface is not available, thus the functionality of the EPC can not be enabled to legacy and Web-services (e.g. REST or SOAP services). In order to tailor better these types of operations and to reduce the overhead signalling and data required, the S3C will execute an interpretation and a translation of the specific service requirements and will interact directly with the core network accordingly.

Also the S3C can transmit its resource requirements directly to the NetIC Generic Enabler for an optimized handling of the subscriber based functionality at the forwarding level as well as to the CDI-Net enabler as part of the connectivity management inside the mobile devices.

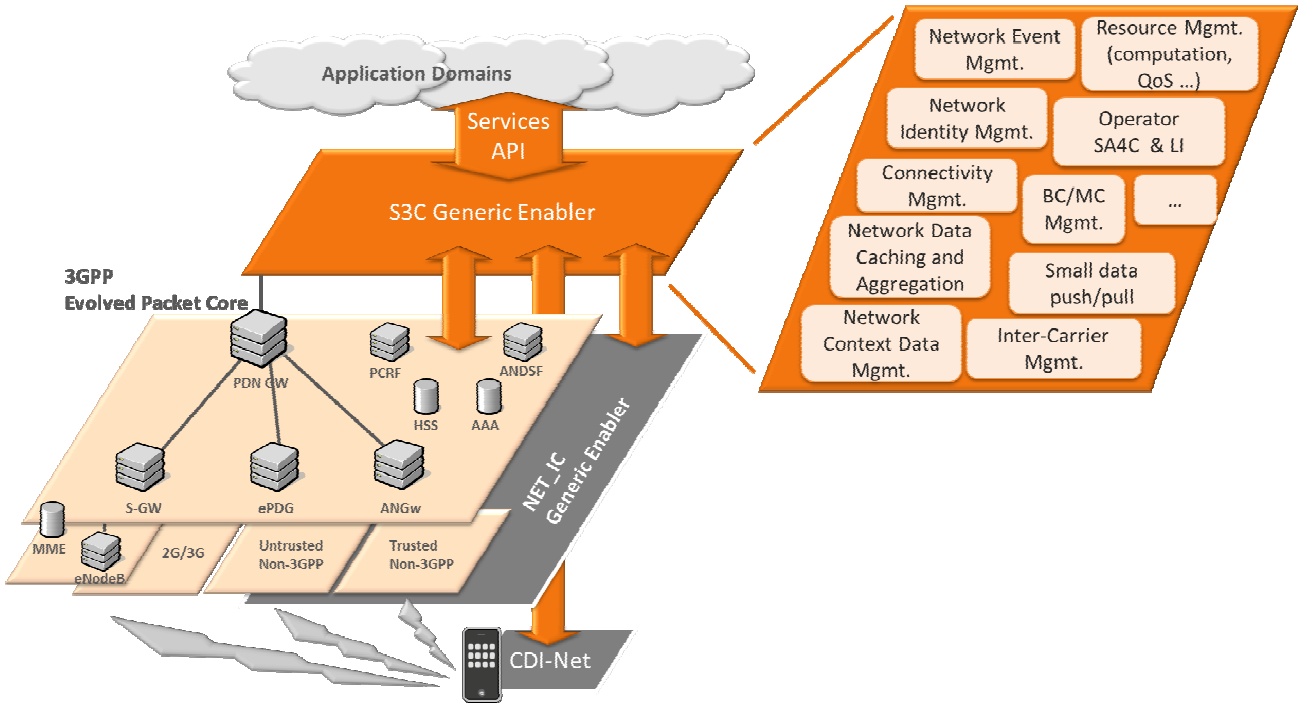


Figure 77 Function blocks within the adaptation layer between FMC control layer and service layer

Functional blocks within the S3C GE will identify the different needs for “translation”:

- Network Event Management (NEM)

  The NEM function of the S3C Generic Enabler provides to the different services mechanisms for subscribing and receiving notifications on data path events related to the mobile devices.

  The events, to which a service platform subscribes includes the loss of connectivity from the mobile device, the access network change, modification of the charging characteristics etc.

  Through this mechanism the services are able to receive information which enables the establishment, adaptation or smooth termination of the active sessions.

- Resource Management (RM, e.g. Computation, QoS)

  For the various services to be able to communicate with the mobile devices, the S3G Generic Enabler includes a Resource Management function which enables the aggregation and convergent presentation of the applications requirements towards the core network.

  The functionality of the Resource Management includes the management of the required resources at specific locations for the mobile device, the mobility management and the access network discovery and selection enabling the optimization of the usage of the core network resources for various purposes such as load balancing, scalable usage of the network resources, access network and data traffic offload enabling an efficient network support for services such as computation offload which require low delay and high throughput for short time intervals.

  The Resource Management sends these requirements dynamically, based on the connectivity status of the mobile devices to the control entities of the core network enabling it to establish the parameters of the communication of the mobile devices accordingly.

- Network Identity Management (NIM)

  The NIM functionality of the S3C GE allows the secure management of identities in the network. Hereby, identities refer not only to user identities, but also to identities of devices, services, applications or end device functionalities and groups of such. This puts users and services in the position to verify the identities of each other, so that a user/device/service can be sure to perform transaction with the correct peer.

  Services are enabled to create and manage network level groupings of identities for optimal communication purposes, e.g. exploit network multicast building blocks. The NIM also provides methods to authenticate identities, e.g. by use of cryptographic methods.

  Identities are associated with different attribute sets, according to the context in which they appear. E.g. an online shop application may only retrieve the relevant attributes, belonging to that applications context.

  The NIM also supports S3C internal charging and billing by resolving identity groups to separate chargeable identities, related to service users, but also to service providers.

  Last but not least, these functions have to coexist with and/or provide support to the functions provided by Identity Management GEs in the FI-WARE Security chapter.

- Operator SA4C (OSA4C) including Legal Interception (LI)

  The OSA4C function will deal with different sub functions:

  - Security – in the sense of encryption and decryption algorithms and of keying and key management.
  - Authentication, Authorisation, Accounting, and Charging – Authentication and Authorisation are responsible to get access, whereas Accounting in Charging ensue to fulfil the possibility to perform business aspects for an operator.
  - Auditing – will offer the possibility to keep track with the business transfer.

  Accounting and in auditing mechanisms are also the base for Legal Interception. Which data and which aggregated state will be provided to governmental institutions are defined in each country separately, even so in a globalised market it would be good to come to a common few.

  The challenge is to implement a system function to optimise the access to third party providers as well as not to loose the control for the network service provider, which own the virtual or real infrastructure. This requires analysis of how this functions can coexist with GEs defined in the FI-WARE Security chapter.

- Connectivity Management Entity (CME)

  The CME is responsible for monitoring and controlling the attachment of mobile devices. In other words, the CME controls via which access networks/interfaces a mobile device connects to the core network.

  Monitoring involves maintaining a list of available access nodes for every mobile device. An available access node is a node to which the mobile device could potentially set up a connection. Monitoring also involves keeping track of the mobile device states and the access node status, e.g. the number of connected devices.

  Controlling involves initial configuration and association of an interface to an access node. In principle it can be distinguished between single technology devices and multi technology devices. In the first case the CME controls to which access node of the available technology the mobile node should connect to. In the second case the CME controls which access node of which technology to use for a specific service. This could also mean that multiple interfaces (technologies) on a single device can be active at the same time to serve different applications.

  In addition to that the controller could also initiate handovers (horizontal and vertical) from one access node to another.

The CME also has an interface to the Resource Management function in order to base its decisions on current resource situation in the network.

- Multicast/Broadcast (MC/BC) Management (MC-BC-M)

  In an end-to-end communication pattern, many networks and network types are typically involved. This starts with the RAN at the mobile terminal or other last-mile network for DSL, then goes through the access and core networks until it either reaches a service or another mobile/fixed terminal. These networks can offer specific properties for MC or BC communication within their own scope.

  The MC-BC-M function has the purpose to exploit such properties and to enable the usage of such to services in order to allow an optimized communication, especially towards fixed/mobile terminals, e.g. by exploiting the broadcast capabilities of a RAN.

  The main use case is optimized real time video broadcasting to mobile terminals, but also group information services.

- Network Data Caching and Aggregation (NDCA)

  Parameters from (mobile) terminals and network nodes close to the terminals will over a huge range of information to optimise the network, application, and service usage in the network as well as in the devices and the cloud.

  In special cases, it is not necessary to access the terminal which acts as an information source directly. Such cases are, when there was no change in the network context information. This information can be stored and cached in the node close to the terminal. So that spare resources – e.g. the air interface or the energy in a terminal is low – will not be used and the information can be requested from these nodes (e.g. a base station). The node is also responsible to aggregate the information and to provide a full information packet to the requesting network instance.

  The challenge is to use existing protocols to build and manage such an environment and define the proper information data set and interface to transfer the information to the cloud infrastructure.

- Small Data Pull/Push (SDPP)

  SDPP messages are required for various applications, e.g. to request new data or to notify a specific service (running on the device or in the cloud) about events. Such messages are very small in size compared to ordinary messages. SDPP aims at providing data channels that can be used to transmit small messages (e.g. a few bytes) to a specific application.

  There are existing concepts that allow for SDPP (e.g. Apple Push Notification Service, Android Cloud to Device Messaging). However, they utilize plain TCP connections and involve a lot of unnecessary overhead (e.g. keep alive messages, radio module needs to stay powered on etc). The latter problem is caused by the plurality of applications, which run in the background and frequently request or receive information from distant servers.

  The challenge for SDPP is to design an interface at the networking layer that accepts push and pull request from multiple sources and delivers them efficiently by taking into account the characteristics of the wireless carrier. On the other hand, the ways in which this capability can be exploited in the implementation of Publish/Subscribe Broker GEs as defined in the FI-WARE Data/Context Management chapter have to be explored.

- Network Context Data Management (NCDM)

  Context data comprises information such as location, device status or user activity, but also network context data that is derived from nearby network nodes.

  The NCDM aims at collecting and provisioning such data to mobile services and applications as well as Internet services and applications. The challenge related to NCDM is to define an appropriate interface that enables the access to the context data and allow e.g. to subscribe to certain context- and location-based events. The way in which these functions link to Data/Context Management GEs in FI-WARE, have to be analyzed.

- Inter Carrier Management (ICM)

  The Internet is an accumulation of different Internet network service providers, which run their business in a quite independent way. Even in the current Best-Effort basis, there are some Service Level Agreements in place.

  Since the service request for the Internet will become in the framework of Fixed-Mobile-Convergence highly dynamic, we need mechanisms in place to support these dynamics.

  Currently a lot of effort is done in the framework definition, optimisation, and integration of dynamic and QoS based peering. The necessary protocols will be used and extended to the needs of the Use Case projects and other potential Use Case areas.

7.2.4.2 *Critical product attributes*

- Based on the philosophy of Future Internet, Fixed-Mobile-Convergence, and all-IP communication, S3C GE will bundle – and if needed – extend existing standards and implementations to manage and control all up-coming services and applications independently on top of a heterogeneous network infrastructures and multi-provider domain scenarios including connected third party provider, content providers, and cloud infrastructure providers.
- In details, the S3C GE will provide a unique convergence layer towards the current and future communication world by implementing interfaces for the operator driven network infrastructure (SIB-driven), and for the Web and Cloud community services (REST and SOAP-driven), as well as towards the rest of applications and services which have no defined APIs and interfaces .
- S3C GE will enable a higher level of scalability for the core network operations through customization on a subscriber base with the goal of enabling a higher number of devices to connect and to access their services in an efficient manner without requiring any more physical and functional extensions.
- S3C GE will include certain interfaces and APIs for business and legal functions and processes.

7.3 Question Marks

7.3.1 Security aspects

Identity & Privacy Management

Management of Identity of user and definition of rules to provide access to the data in a secure and controlled way are mechanisms expected to be provided for connected devices, cloud proxies, as well as network connections, services and applications running on top of the FI-WARE platform. It however necessary a detailed analysis of the best solutions to be applied through the GE functionalities.

7.3.2 Other topics still under discussion

Interoperability of Generic Enablers

There are functionalities that, at first sight, might be considered as possible superposition among the Generic Enablers. While this is an issue to be surely addressed in some situation where the complete and

precise functionality is not yet fully explored (this will be one important activity in the next project period), in general the apparently similar functionalities are rather inter-operation functionalities among the different GEs.

This is the case of CDI and S3C Generic Enablers; both GEs provide functionalities concerning network connectivity, where:

- CDI offers management info/policies which the device uses to better manage on its own connectivity parameters (e.g. when to attach to a network, to which interface to forward a data packet) which are further exposed to applications,
- S3C offers a control of the communication operations (e.g. how the data pipe is established over a connection of the device).

On the other hand, the interconnection and its flow among the GEs (unidirectional/bidirectional), as shown in Figure 66, is based on the current definition of the functional elements belonging to each GE. It is expected that, due to further requirements coming in the future (e.g. from Use Case projects) such interconnections will be updated according to the emerging needs. The same will apply of course for the inter-operability with all the other GEs of the FI-WARE architecture.

7.4 Terms and definitions

- **Connected Devices:** A connected or smart device can be an advanced device located at home, such as a set top box and multimedia device (including advanced TVs), PCs, storage (NAS like), indoor handset (home/advanced DECT), or game consoles. Furthermore, mobile devices, such as mobile/smart phones (GSM/3-4G), tablets, netbooks, on-board units, (in-car devices) or information kiosks are connected devices, too. It is very likely that new devices will appear and fall in this “smart devices” category during the project execution (femto cells, etc.).
- **Cloud Proxy:** A device encompassing broadband connectivity, local connectivity, routing and networking functionalities as well as service enabling functionalities supported by a modular software execution environment (virtual machines, advanced middleware). The “Cloud Proxy” or “Home Hub” is powerful enough to run local applications (for example home automation related tasks such as heating control or content related ones such as Peer to Peer (P2P) or content backup). It will also generally include local storage and may be an enabler for controlling privacy as some contents or data could be stored locally and could be controlled only by the user without having the risk of seeing his/her data controlled by third parties under consideration of the overall security architecture.
- **Open Networking:** Open networking is a concept that enables network nodes to provide intelligent network connectivity by dynamic configuration via open interfaces. Examples for provided features are the fulfilment of bandwidth or quality requirements, seamless mobility, or highly efficient data transport optimised for the application (e. g., with minimum network resource or energy consumption).
- **Network Service:** Network Service is a control and policy layer/stratum within the network architecture of a service provider. The Network Service provides access to capabilities of the telecommunication network, accessed through open and secure Application Programming Interfaces (APIs) and other interfaces/sub-layers. The Network Service concept aims at providing stratum that serves value-added services and applications at a higher application and service layer and exploits features of the underlying transport and technology layer (e. g. by NetIC interfaces).

7.5 References

[IPsphere]	IPsphere project within the Telemanagement Forum, http://www.tmforum.org/ipsphere
[StrataModel]	T. Nolle, "A New Business Layer for IP Networks", July 2005 Issue of Business Comunications Review, 999 Oakmont Plaza Drive, Suite 100, Westmont, IL 60559, 630/986-1432, www.bcr.com – http://www.ipsphereforum.org/Files/A New Business Layer for IP Networks –-TN1.pdf
[TMF]	Telemanagement Forum, http://www.tmforum.org/
[Cube01]	R. Aguiar, H. J. Einsiedler, J. I. Moreno, "An Operational Conceptual Model for Global Communication Infrastructures", Wireless Personal Communications – Global Information Multimedia Communication Village (GIMCV), Volume 49, Number 3, May 2009, Springer Press Netherlands, Selected topics from the Strategic Workshop, May 28-30, 2008, Madeira, Portugal , ISSN 0929-6212 (Print) 1572-834X (Online).
[Cube02]	R. Aguiar, H. J. Einsiedler, J. I. Moreno, "A Requirements Analysis for the Protocol Stack of the Future Internet", International Conference on Communications 2009 (IEEE ICC 2009), International Workshop on the Network of the Future 2009, 18 June 2009, Dresden, Germany.

8 Security

8.1 Overview

Future Internet is expected to implement what is commonly called as a “Internet of Services”(IoS), a generalized service-oriented architecture where services and service providers will collaborate and compete.

In this IoS, the necessity of individualization will require the capacity to guarantee that the personal information provided by users will be processed in accordance with the user rights and requirements. As a common example, let’s say you want to create a Multimodal travel made easy, online services offer a traveler a wide range of travel and transport options, according to the user’s preferences, for all the stages of a trip that may use various modes including public transport, car, and non-motorized means. In that case new security and privacy functions are required to propagate geo-localization and geo-referencing data facilitating safe and easy rendezvous between drivers and travelers, payment information allowing proportional automatic contribution to journey or specific security functions ensuring the safety of all participants through a careful set of preventive, en-route and forensics functions.

Future Internet services will always be exposed to different types of threats that can lead to severe misuse and damage. Creating safe and trusted services without sacrificing much of the desired functionality, usability, performance and cost efficiency is a great challenge, especially in a dynamic environment where new threats and attack methods emerge on a daily basis.

The Future Internet will provide an environment in which a diverse range of services are offered by a diverse range of suppliers, and users are likely to unknowingly invoke underlying services in a more dynamic and ad hoc manner. Moving from today’s static services, we will see service consumers that transparently mix and match service components depending on service availability, quality, price and security attributes. Consequently, the applications the end-users sees may be composed of multiple services emanating from many different providers, and the end user may have little in the way of guarantee that a particular service or service supplier will actually offer the security that they claim.

In this context it becomes essential to have means of security monitoring extremely efficient and respond quickly to attacks. Terrorist groups express their objective to unleash cyber attacks that are harder to detect and defend against them. Indeed, future acts of terror may come not only from suicide bombers wearing explosives belts but from a few keystrokes on the computer: a weapon of mass disruption. Of course, the Security monitoring covers more common threats, like toll-fraud, impersonation, service high jacking.. To defend ourselves, Future Internet services need more intelligent early attack detection and support for decision and rapid action making faced with constantly evolving threats. This is one of the challenges of FI-WARE.

The current landscapes of service delivery ecosystems do not fully address principals such as openness, usability and simplicity. FI-WARE aims to balance between simplified service usage and end user trust (including underlying security) in the service. FI-WARE will be designed in a flexible manner in order to reflect generic as well individual requirements. By that FI-WARE will be easily adaptable to upcoming needs. Furthermore this also is supported by including social interactions being part of the working community, e.g. by offering a “security market place” where anyone interested could contribute. A typical example of such a marketplace can be the sharing of vulnerable configuration descriptions within a community of users, allowing faster reactions and even prevention from potential attacks exploiting these vulnerabilities.

The overall ambition of the Security Architecture of FI-WARE is to demonstrate that the Vision of an Internet that is "secure by design" is becoming reality. Based on achievements to date and/or to come in the short-term (both from a technological but also a standardization perspective) we will show that "secure by design" is possible for the most important core (basic) and shared (generic) security functionalities as

anticipated by the FI-WARE project and in accordance with the requirements of external stakeholders and users such as the FI PPP Use Case projects. The "secure by design" concept will, therefore, address both the security properties of the FI-WARE platform itself and the applications that will be built on top of it. As such, the Security Architecture will focus on key security functionalities such as identity management or security monitoring to be delivered as so-called generic security enablers that will be integrated with the design and implementation of the FI-WARE. The basic security architecture will be designed to be extensible to meet additional security requirements coming from both the FI-WARE project (development and research activities, market analysis and consortium-specific exploitation requirements) and the FI PPP Use Case (UC) projects and their trials.

Security, Privacy and Trust in FI-WARE will be mainly focusing on delivering tools and techniques to have the above-mentioned security needs properly met. This will be performed by design and some semi-automation and assistive technology to alleviate the workload of users and administrators while raising their security awareness to make informed decision.

In this section the foreseen high-level functional architecture is described, introducing the main modules and their expected relationships, then depicting the most important modules in detail along with their main functionalities.

The high level architecture is formed by four main modules: Security monitoring mechanisms (M1), a set of General Core Security Mechanisms (e.g. Identity Management and Privacy solutions) (M2), Context-Based Security and Compliance (M3) where an enhanced version of USDL for security will support the matching of security goals with available security services while addressing compliance management, and a set of universally discoverable Optional Generic Security Services (M4) that will be instantiated at runtime and can be dynamically reconfigured (triggered by M3) based on the needs of specific scenarios.

The overall security plane of the FI-WARE architecture will interlink with practically all its functional modules. In order to simplify the description of these links subsequently the main components as well as their technical relationships with only the Application and Service Ecosystem and Delivery Framework and FI PPP Use Case projects are depicted:

The core general security mechanisms for the FI-WARE project will be provided by M2, including support for Identity Management, Authentication Authorization and Access, and Privacy. M3 will provide the required language and tools for describing services in the FI and their security needs. Where specific scenarios will require optional generic security services these can be consumed on a basis of what is provided by M4. A key architectural assumption is that security services may fail. Security monitoring mechanisms as provided by M1 may detect deviations with respect to the expected behaviour and signal this to M3 to take action (e.g. invoke alternative security services or trigger countermeasures if under attack).

FI-WARE GEs to be developed and/or integrated as part of the Security chapter will materialize the (Security) Reference Architecture sketched in Figure 78. This Reference Architecture comprises:

- A component able to dynamically invoke and compose security services to answer related security needs while dealing with constraints which may apply (e.g. regulatory).
- A set of GEs for a number of shared security concerns (i.e. identity and access management as well as privacy and auditing) that are considered core and therefore present in any FI-WARE Instance.
- A set of optional Security GEs to address current and future requests from concrete Usage Areas.
- An advanced security monitoring system that covers the whole spectrum from acquisition of events up to display, going through analysis but also going beyond thanks to a digital forensic tool and assisted decision support in case of cyber attacks.

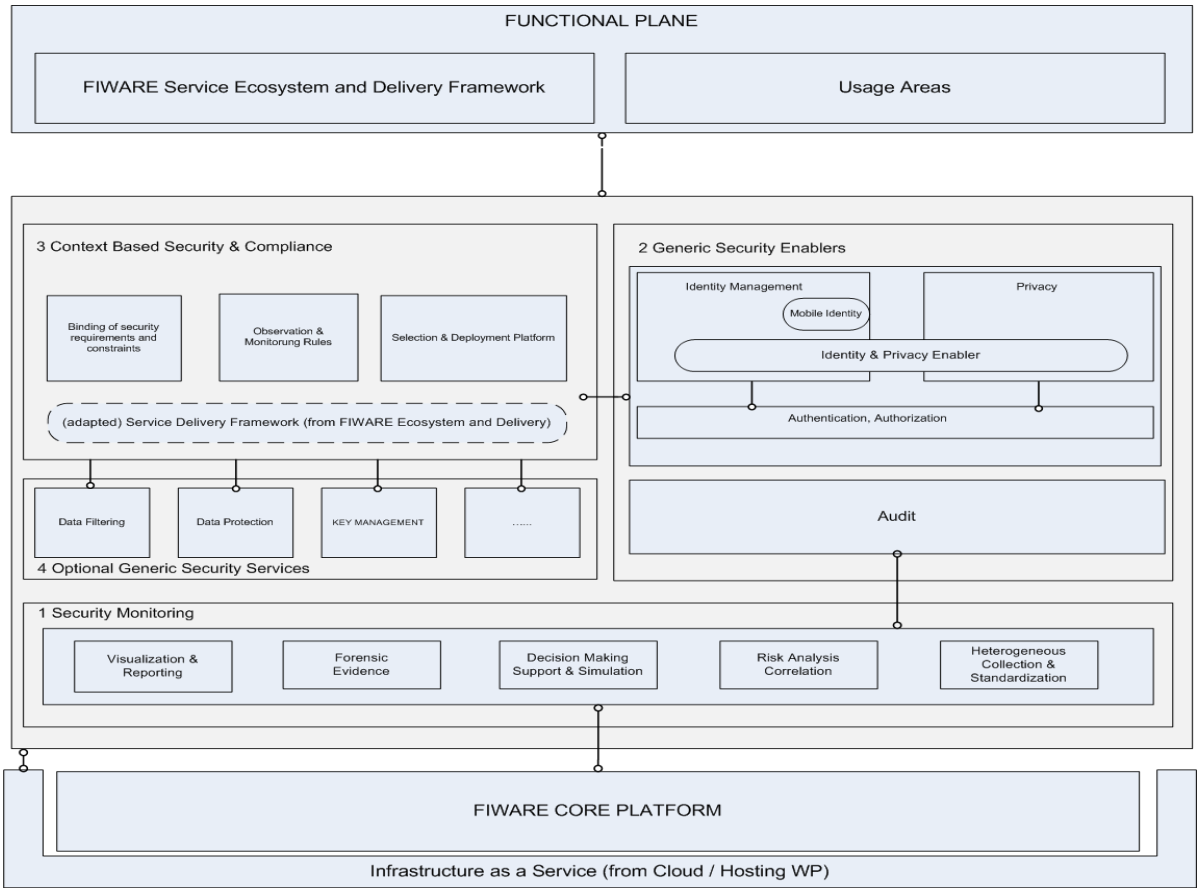


Figure 78 FI-WARE High Level Security Architecture

8.2 Generic Enablers

8.2.1 Security monitoring

8.2.1.1 *Target usage*

The Security Monitoring GE is part of the overall Security Management System in FI-WARE and as such is part of each and every FI-WARE instance. The target users are: FI-WARE Instance Providers and FI-WARE Application/Service Providers.

Security monitoring is the first step towards understanding the real security state of a future internet environment and, hence, towards realizing the execution of services with desired security behaviour and detection of potential attacks or non-authorized usage.

Security monitoring is focused essentially on monitoring alarms from network equipment, systems and security sensors. By the collection, filtering and correlation of data from large-scale heterogeneous environments, including sensitive data from security tools and devices, SCADA events, raw sensor data, suspicions behaviours, etc., coupled with a dynamic risk analysis engine, decision making support and role-oriented visualization engine, the security stakeholders can take appropriate actions to prevent and mitigate the impact of abnormal behaviour.

In addition, the availability of digital forensic evidence models and tools will provide a digital forensic for evidence solution to analyze abnormal behaviour, carry out a criminal investigation and provide evidence with legal value.

8.2.1.2 *GE description*

Overview

The GE as envisaged will address security monitoring and beyond, up to pro-active cyber-security i.e. protection of “assets” at large. The figure below provides a high-level initial architectural sketch of the Security Monitoring GE as envisaged in FI-WARE. It shows both how it is structured and how it is expected to work. The targeted functional components associated to this GE are described in the following subsections.

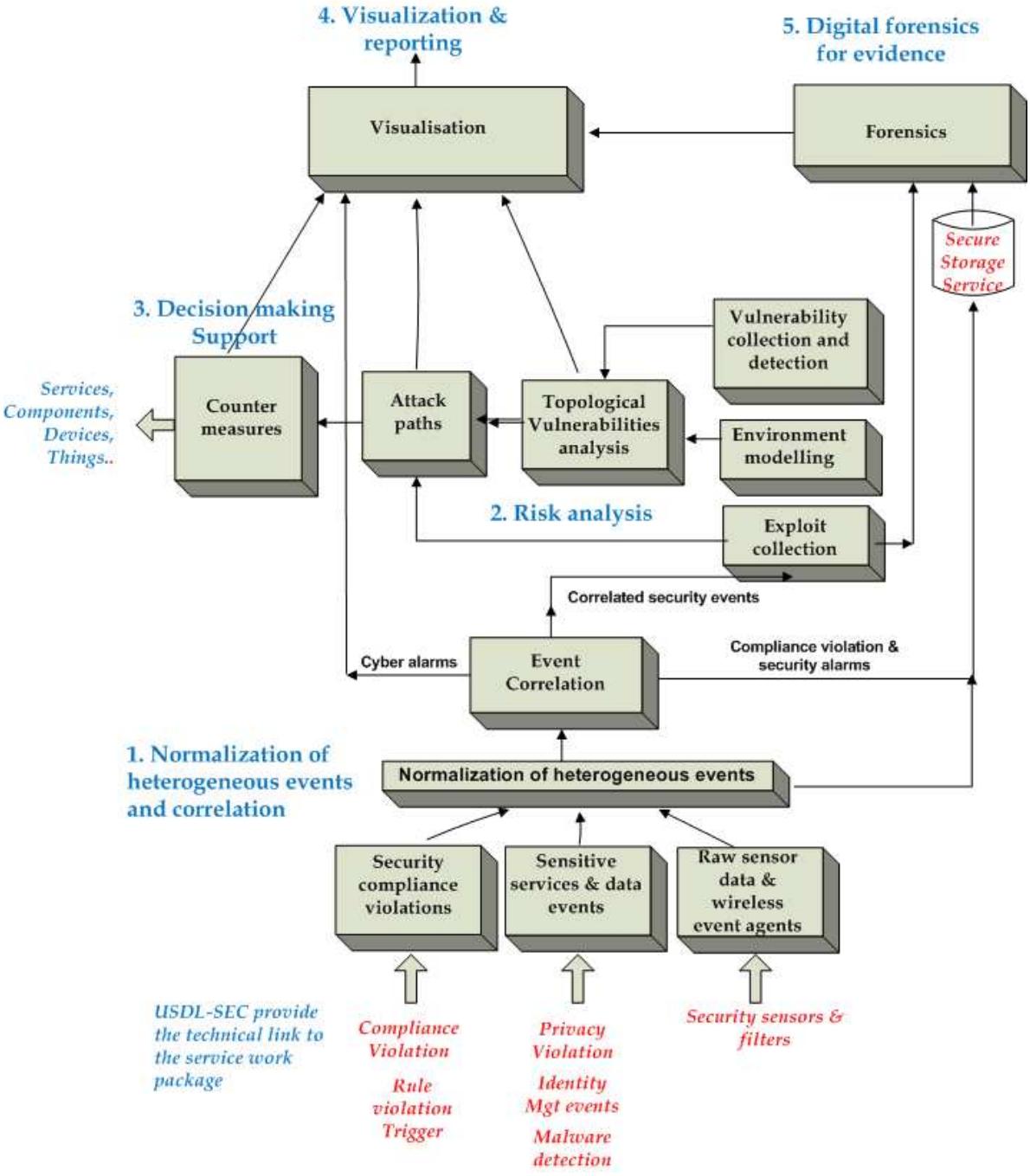


Figure 79 Security Monitoring GE

Normalization of heterogeneous events and correlation

This component is shown as [1] in Figure 79. It will cover the following functionalities:

- **Massive security events collection** e.g. Firewalls, Intrusion Detection Systems, Security Information and Events Managers, (non-exhaustive list)

- sensitive data events, Things events, FI-WARE service events,
- raw sensor data & wireless event agents,

- Normalization and correlation of security events

  The Monitoring Security Enabler will exploit the security events logged by FI-WARE services and applications (i.e. non-Authorized access attempts, service disabling, denial of service attempt..). We will exploit them to detect an intrusion. Event correlation technology has been available in the monitoring field for years, but until now it has been based on static rules which are very hard to create (several sources of events), update (thousands and thousands of event identifiers) and understand (identifiers are just numbers). The proposed functionality employs artificial intelligence algorithms to create correlations based on tags instead of identifiers, which are much easier to understand and allows independent sources of events. Changes in the pattern of the attacks do not require major updates of the algorithms.

- Anomaly detection

  Anomaly detection is a monitoring technology that complements signature-based intrusion detection, based on searching for deviations from a normal behaviour model. It helps in detecting 0-day malware, encrypted attacks and in general, all those attacks that use common protocols such as http, SMTP, etc and, thus, are invisible for the majority of monitoring systems.

  The Behaviour Anomalies Network Detection System in FI-WARE goes beyond statistical and aggregated anomaly detection, and uses artificial intelligence algorithms to learn the normal behaviour of each user. Behaviour Anomalies Network Detection System enables the creation of several behaviour models per user.

- Security compliance violation alarms
- Attack detection alarms

Risk analysis

This component is shown as [2] in Figure 79. It will cover the following functionalities:

- Vulnerabilities collection and normalization

  The Open Vulnerability and Assessment Language (OVAL) and the Information Systems Threat analysis Open Product (ISTOP) solutions are designed to contextually evaluate and manage the vulnerability level of Information Systems.

  The OVAL language standardizes the three main steps of the assessment process: representing configuration information of systems for testing; analyzing the system for the presence of the specified machine state (vulnerability, configuration, patch state, etc.); reporting the results of this assessment. The repositories are collections of publicly available and open content that utilize the language.

  The vulnerabilities collection is used by the topological vulnerability analysis.

- Exploits collection

  An easy to navigate exploits database enables the generation of automated multiple attack scenarios, utilizing vulnerability collection to determine the success of an exploit series.

- Attack penetration modeling

  A series of exploits is an attack path and the set of all possible paths of attack is an attack graph. Nodes and arcs represent actions the attacker takes and changes in the network state caused by these actions. The goal of these actions is for the attacker to obtain normally restricted privileges on one or more target hosts.

- Attack path identification

  The objective is to preemptively identify the attack paths. These attack paths will be identified by simulation of advanced exploits, with these exploits taking account of the vulnerabilities of the

monitored environment. The discovery of these paths requires, a priori, a vulnerability topological analysis and the environment modeling. Normalized vulnerability collection (i.e. coming from CERTA collection) as well as a collection of the exploits most frequently used by the hackers will be used.

- Business risk impact evaluation

  Of course, it's not easy to evaluate the business impact without to mention urbanization map or complex business processes, but it is possible to identify critical services and sensitive data and to establish some priorities and adapted countermeasures. Business risk management includes the methods and processes used by organizations to manage risks and seize opportunities related to the achievement of their objectives. Business risk management objectives cover two main circumstances:

  - adopting measures to prevent situations that can put the organizations business at risk, such as
    - Determine the common security level of the computer network or system under protection and calculate a set of security metrics describing different aspects of security on various levels
    - Detect vulnerabilities and errors in security services/mechanisms/policies and network configuration that reveal possible assault actions for different security threats
    - Determine critical network or system resources and choose effective security policies appropriate to threats
  - actions to identify events or circumstances relevant to the organization's objectives, assessing them in terms of likelihood and magnitude of impact, determining a response strategy, and monitoring progress. The methodologies of business risk analysis are able to:
    - Detect risks affecting the realization of company objectives and non-compliance with regulatory requirements.
    - Map the risks identified to discrete business processes in order to obtain a list of the business processes to be considered.
    - Measure risks according to the probability of the event occurring and the impact of the event and taking into account the consequences of a loss of confidentiality, integrity and availability.
    - Determine whether the level of risk is acceptable or requires mitigation.
    - Obtain management buy-in for the project and assign roles and responsibilities as well as a timeline for the implementation

  Both processes should provide a reasonable assessment if an organization is achieving its high-level objectives and regulatory requirements like:

  - Operational and strategic objectives (including the improvement of performance),
  - Effective reporting (accurate and reliable),
  - Compliance with laws and regulations.

Decision making support and countermeasures

The Security Monitoring GE support manual or semi-automated selection of countermeasures (shown as [3] in Figure 79) mitigating cyber-attacks and reduces the risk level. This functionality is capable of proposing the most appropriate mitigation responses based on results of the risk analysis functionality.

Countermeasures can involve business experts, security and process managers, and watch keepers. It is therefore highly beneficial to accomplish a risk assessment, the goal of which is to weigh uncovered threats based on their probability (influenced for instance by the preparedness of a potential attacker), impact (e.g. by estimating the costs of a successful attack) and some additional factors (like the costs of the possible countermeasures). Such an assessment will help finding the most appropriate balance between security level and the associated costs.

Visualization and reporting

The Security Monitoring GE will provide a dynamic, intuitive and role-based User System Interface (shown as [4] in Figure 79) for the various stakeholders to use in order to understand the current security situation, to make decisions, and to take appropriate actions. It will support the following features:

- Multi-view and multi-perspective user interface, able to present a high-level business view, illustrating the impact on the current business operation, as well as lower level, more technical, views such as at the system or network level.
- Rapidly customizable views to enable the user to select the data and information they need to see, along with the visualizations they find most effective. There are many potential visualizations, some of which are more suited for users in some roles than others, and indeed users will find some easier to work with for them personally compared to other users in the same role. The interface must enable the user to rapidly and easily select and arrange visualizations as they see fit.
- Rapid addition of new data and information sources as well as visualizations. In order to assess the likely dynamic and complex security situations of the Future Internet, the ability to rapidly add and visualize new sources of data and information as they arise will enhance the user's ability to make decisions.

Digital forensics for evidence

The high-level process of digital forensics (shown as [5] in Figure 79) in the Security Monitoring GE deals with the acquisition of data from a source, the analysis of the data and extraction of evidence, and the preservation and presentation of the evidence. The digital evidence is intended to facilitate the reconstruction of events found to be malevolent or helping to anticipate unauthorized actions

Digital evidence, by its very nature, is fragile and can be altered, damaged, or destroyed by improper handling or examination. For these reasons special precautions should be taken to preserve this type of evidence. Failure to do so may render it unusable. Original evidence will be acquired in a manner that protects and preserves the integrity of the evidence. A protection will be initiated to preserve and protect original evidence.

Correlating events provides the means to support the search for evidence process. Timeframe analysis can be useful in determining when events occurred. For this, we can review the time and data stamps contained in the file system metadata, linking error logs, connections logs, security events, alarms and files of interest to the timeframes relevant to the investigation.

Analyzing every bit of data is a daunting task when confronted with the increasing size of storage systems. A second problem is that acquired data are typically at the lowest and most raw format, which is often too difficult for humans to understand. The purpose of digital forensic analysis tools will be to accurately analyse data forensic collections at a layer of abstraction and format that can be effectively used by an investigator to identify evidence.

8.2.1.3 *Critical product attributes*

Compared to existing products and services, the security monitoring service as targeted in FI-WARE, offers a unique selling point. Today, no comprehensive security monitoring service ranging from network to the application exists for the Internet and applications running on top of it. Of course there may exist some products that could be used either in isolation or in conjunction but none of them has been integrated by design nor would offer the level of functionalities targeted here and described above.

Furthermore apart from satisfying the needs identified, the security monitoring service of FI-WARE will also be targeting some unique features not only demanded but also key to its success and wide adoption, including:

- Integrity of the security monitoring service (applies to all security services)

- Service usability and intelligibility,
- Performance,
- Pro-active cyber-security enablement
- Assisted and informed decision making in case of alarms raised (e.g., events anticipating attacks, events reporting compliance violations)

In summary the Security Monitoring service of FI-WARE seen as GE according to the definition shared and agreed by project team offers several unique features:

- A comprehensive and pro-active security monitoring system ranging from acquisition of events to role-oriented display, providing "intelligibility" of the visualization according to each of the stakeholders' perspective)
- Business-oriented decision making support (in case of cyber threats).
- Digital forensics tool, also targeting at Services
- Facilities to promote Collaborative Security (thanks to the openness of the GE able to serve other security-related purposes (e.g. normalized events could be shared with other EU Security Operational Centers to foster Collaborative Security at EU level)
- Better Trust and Confidence in using Future Internet Services (main benefit for the EU Citizen)
- Significant aid to fight cyber crime which today's seriously impede the EU Economy.

8.2.2 Identity Management

8.2.2.1 *Target usage*

This enabler provides authentication and identity/attribute assertions as a service to relying parties. The relying parties are typically service providers that provide easy access to their services to users by means of single sign-on (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 which also protects the users' privacy. The Identity Management core 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 OpenID [OpenId] and OASIS SAML v2.0 [Saml] (Security Assertion Markup Language). Motivated by the IoT, the enabler also covers new user attributes such as things, as well as it manages the identity of things themselves (attributes, current users, location, use history, etc). The large number of sensors and mobile devices poses new challenges; identity federation and single-sign-on support ease of use.

Furthermore, the authentication feature of the enabler also covers the authentication of things for services, other objects or users as relying parties, and the authentication of users, services and other things for things as relying parties. It also supports user SSO across multiple things. Motivated by Cloud computing, the enabler can be run in the cloud as well; when doing so, special care is taken so that the sensitive data is not exposed to the threats related to the nature of clouds (e.g. deployment in a public cloud).

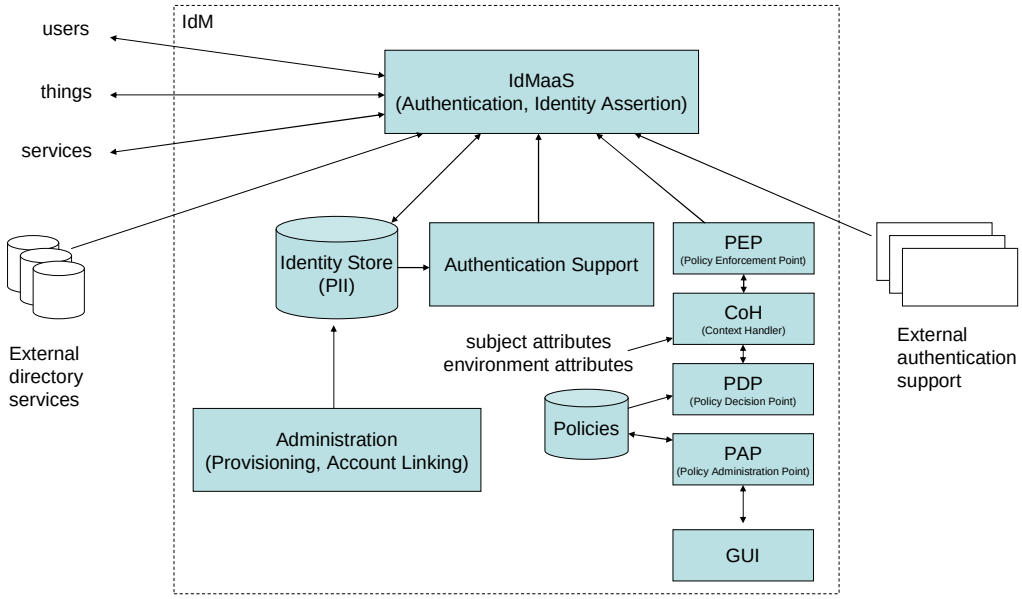


Figure 80 Identity Management core functional architecture

8.2.2.2 *GE description*

IdMaaS is the "front end" of the IdM solution; it communicates with the relying parties (services, things) as well as with the subjects of authentication and attributes assertions (users, things). Given the heterogeneity of the FI Core Platform, identities are usable across trust domains. Hence identity translation services (Secure Token Services) are expected to be a key component of the IdM solution.

The Authentication Support connects the front end to the underlying authentication machinery where applicable; e.g. beyond the simplest password-based authentication of users, it can support the re-use of the network/telco-level authentication of users or things at communication services providers (telco operators) by means of GAA/GBA [GaaBook08] or by connecting to the Authentication-Authorization-Access function of the access network.

The Identity Store is a database of identities including identifiers and other attributes. It can be complemented with external stores as well.

The Administration elements provide access to the identity data for administration purposes.

Privacy protection is done by means of standard XACML [Xacml] machinery. The requests for PII (personally identifiable information) are intercepted and controlled by a Policy Enforcement Point (PEP). The PEP hands the request over to the Context Handler (CoH) in its native request format, optionally including attributes of the subjects, resource, action and environment. The CoH constructs an XACML request context and sends it to the Policy Decision Point (PDP). The PDP requests any additional subject, resource, action and environment attributes from the context handler. The PDP evaluates the applicable policies and returns the response (authorization decision) to the PEP. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access. For a more detailed description, consult the XACML specification. And important part of the solution is the GUI (graphical user interface) by which end-users can manage their own privacy policies.

8.2.2.3 *Critical product attributes*

- Authentication and attribute assertions about users and things as a service to relying parties via open protocols (OpenID, SAML).
- Single sign-on (SSO) for end-users.
- Enforcement of end-users' privacy policies in attribute assertions.
- Extensibility with external identity stores and authentication support functions.
- A graphical user interface (GUI) for end-users to manage their privacy policies.

8.2.3 PrimeLife Policy Language (PPL) Engine

8.2.3.1 *Target usage*

This enabler supports the Identity Management core generic enabler as well as other attribute provider services in respecting the end-users' privacy preferences.

8.2.3.2 *GE description*

This GE supports the Identity Management core GE as well as other attribute provider services in fulfilling the end-users' privacy preferences.

End-users should be provided with strong control over the disclosure and use of their personal data by application/service providers and devices. The technical solution guarantees the correct enforcement of the privacy policies and regulations.

Currently, websites and online applications acting as data controllers [EuPriv95] are obliged to publish a privacy policy stating how the data collected from users will be handled and treated. This privacy policy is a text is written by layers and most of the time not really easy to understand for the common users. Beside the lack of clarity of such privacy statements, their enforcement is not automated. That means that there is not technical mean to execute the actions and constraints described in such documents. Then it becomes very hard to check whether a data controller is compliant with his declared privacy policy. For instance a user will not be able to verify if the data controller shared his data with a third party. For this reason, we propose to provide a machine readable language called PPL [Ppl] that is able to express the rules contained in the standard privacy policies. This language is not only designed to express privacy policy but also privacy preferences expressed by the users. These preferences can then be compared or matched with the privacy policy of the data controller. The PPL language can express at the same time access control rules (how can access the data and under which condition), and usage control rules (how the data should/must be treated after being collected and for which purpose). Obligations can also been expressed in order to force a data controller to perform an obligation on the data after collecting it (ex. Deletion after a certain period, user notification when the data is used or shared, etc.). The language is symmetric and use similar syntax to express privacy preferences of the user, and the privacy policies of data controller. The agreement between these two parties is expressed into a sticky policy that will travel along with the data in order to keep trace of the applicable privacy rules. This concept is useful when the data is shared and forwarded through various data controllers. At any time the data controller can enforces the privacy rules related to the data.

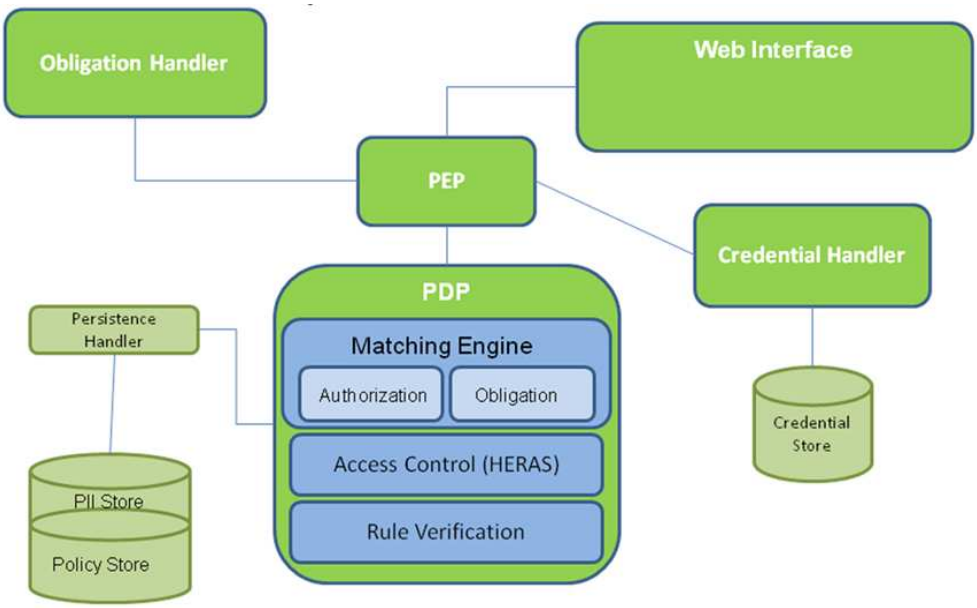


Figure 81 PrimeLife Policy Language (PPL) engine architecture

8.2.3.3 *Critical product attributes*

- Card-based access control with data minimization.
- Integrated access control and data handling: what information needs to be revealed, and how will information be treated.
- Automated matching between user's data handling preferences and server's data handling policy.

8.2.4 Identity Mixer (IdeMix)

8.2.4.1 *Target usage*

This enabler provides private credential systems and minimal disclosure tokens for enhanced privacy protection mechanisms.

End-users are provided with anonymity and the possibility of selectively disclosing their asserted private attributes.

8.2.4.2 *GE description*

Besides the policy-based access control mechanisms included in the previous generic enablers, there are further technology measures for privacy protection. They are usually referred to as Privacy Enhancing Technologies (PETs) and include e.g. interesting and surprising cryptographic solutions such as private information retrieval, zero-knowledge proofs, all kinds of privacy-enhancing signature schemes,

anonymous browsing, secure multi-party computation, etc, though these techniques are still not in wide use at the moment.

One of these privacy-enhancing technologies are private credential systems and minimal disclosure tokens which are a privacy-friendly realisation of attribute based certificates. In a private credential system, users can have different identities with different identity providers and identity consumers. In fact, an identity should be seen as the collection of attribute that are known to a party about the users. Furthermore, identity providers can issue a credential asserting attributes to a user. A user can then selectively reveal the attributes contained in a credential to an identity consumer. That is, the consumer will only learn that the identity provider asserted the revealed attribute but is not able to learn any other asserted attributes. The users can repeat this disclosure process as many times as she wishes without that these transactions can be linked to each (unless the disclosed attributes are unique). Thus, together with attribute-based access control, private credential offer the best possible privacy protection. Finally, we note that the Identity Mixer private credential system also offers all the standard feature of a public key infrastructure such as revocation of credentials or hardware-binding.

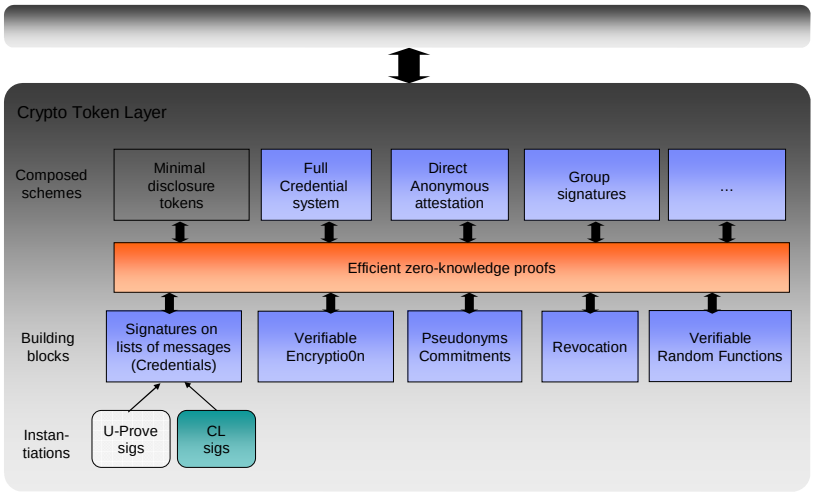


Figure 82 Identity Mixer framework

8.2.4.3 *Critical product attributes*

- Data minimization: only reveal the information that is strictly necessary.
- Optional unlinkability: Different tokens created by the same user cannot be linked -- unless the token was explicitly created in a linkable way.
- Optional untraceability: The user who created a token cannot be identified -- unless the token was explicitly to be traceable by a trusted authority.

8.2.5 Context-based security and compliance

8.2.5.1 *Target usage*

The role of this Generic Enabler is to support additional security requirements requested by a specific subset of applications as a result of the application of very specific regulatory constraints.

The GE will accept security request from a client application and will select the best Optional Security Enabler to fulfil it.

The deployed security enabler will implement the compliance between the client security request and any applicable regulation from Private and or Public sources.

The framework has also monitoring capabilities to overseen the system performance.

As a result of this monitoring, if a non-conformance is detected, the framework is capable of performing run-time and context-based reconfiguration of deployed security enablers, such that the client application will be provided with a new configuration for the security enabler it is utilizing, or it can receive instructions to stop using that security enabler and use a newly provided one.

The figure below provides a high-level initial architectural sketch of the components of this Generic Enabler.

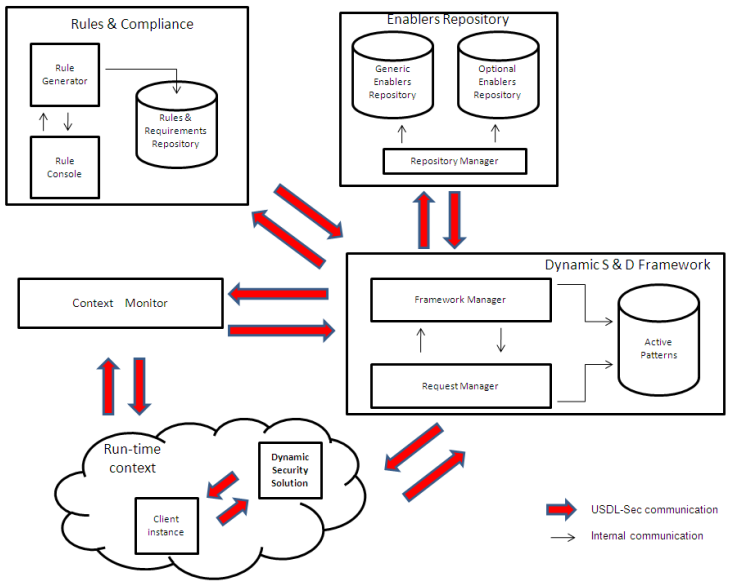


Figure 83 Context-based Security and Compliance

8.2.5.2 *GE description*

The communication between the inner components of the enabler and the client applications that will require its services will be performed by a new standard language to be defined. This new business service oriented language is able to describe & register security services or capabilities.

The main characteristics of this software will:

- Describe functional services provided by the platform and needed by applications
- Describe the interface offered by the security enabler to the applications
- Provide mechanisms to cover from high level description of the service to detail functionalities & implementations.
- Provide event management capabilities to allow monitors get the context event information from the security enablers they overseen.

Extend standard USDL 3.0 by implementing a new module security oriented where security specifications may be referred to existing standards like WS-SecurityPolicy and even management to Common Base Event.

The model is also defining four main functional components which should meet these requirements:

- Rules & Compliance:
  - Support business process modeling capabilities and provide an authority policies generator
  - Ensure that process and policies are compliant (as: implementing Protection Level Agreements)
  - Compliance solutions should be describe in an abstract language and stored in a model repository
  - Provide and store compliance-based metrics to be applied by monitoring components
- Dynamic S&D Framework
  - Run-time & context based reconfiguration of deployed security enablers capabilities
  - On receiving request from client applications it will return an Executable Component that fulfils it
  - The security component provided should implement the compliance between the client security request and any applicable regulation from Private and or Public sources
  - Deployment of an associate monitoring system which will monitor the rules to oversee the compliance of the new context application
  - On non-compliance notifications from the monitoring system S&D Framework will take the necessary actions to provide another security solution dynamically.
- Context Monitor
  - Gets the monitoring rules from the S&D Framework
  - Get then context information from Security enablers
  - Check the applications context fulfils security requirements and any applicable compliance rule.
  - Trigger the framework in case of rule violation
- Enablers Repository
  - Generic and optional enablers already described in this chapter should be register in a security repository to be at the disposal of the framework; witch will deploy them as security solutions to the external applications
  - The enablers should describe their functionalities, interfaces, deployment requirements, compliance capabilities and events

8.2.5.3 *Critical product attributes*

- Provide run-time security support for applications, by managing S&D Solutions and monitoring the systems' context
- Enhanced security and dependability by supporting automated integration, configuration, monitoring and adaptation
- Dynamic compliance of software services to business regulations and user requirements
- Tools for modeling business relations & agreements into an abstract set of rules
- Compliance Repository not only to store, but also managing compliance requirements at these abstraction levels
- A new business service oriented language to describe & register security services or capabilities.
- This language will provide mechanism to cover from high level description of the service to detail functionalities & implementations.

8.2.6 Optional Security Service Enabler

8.2.6.1 *Target usage*

The Security Architecture has also been designed to support the optional generic security services that usage areas can potentially integrate as a generic enabler (so–called ―optional‖ as they are common across many usage areas, but not all). On the basis of a selected set of already available assets, this module will provide use case dependent security services. These optional security services will be described below in this section. Module 4 (in Figure 2) will equally provide the necessary technical means for integration of future security services. The requirements and security goals to be supported will stem from M3, which will enforce the compliant usage of the invoked optional generic security mechanisms and services.

One of the main goals for this asset should be the extension and the consolidation of the USDL-SEC service description language [IoS], in parallel with the work that will be done in M3. USDL-SEC should cover as much as possible the security properties defined in the different generic security enablers. For this task we require inputs from all the modules in the WP8 in order to capture the different security properties of the generic enablers. The optional security services are defined as a modular extension of the main USDL-SEC specification although if the security properties exposed there are domain specific. For this reason we have to define an optional element that can be used to extend the specifications with domain specific security properties such as security plug-ins that can be developed in addition to the basic security features. The second phase of this goal is to describe mechanisms, publish then invoke the optional services as an extension of the basic USDL-SEC framework.

The optional security services do not propose crucial and mandatory security functionalities as provided in M2 (Authentication, Access control, usage control, trust, etc.), but they offer some security capabilities that are applicable for specific cases and scenarios. These optional serves cannot be used by all the applications deployed in FI-Ware. Each optional security service is independent from any other generic security service and not restricted to a specific application domain. The usage of these services must be as easy as possible with no strong configuration requirement.

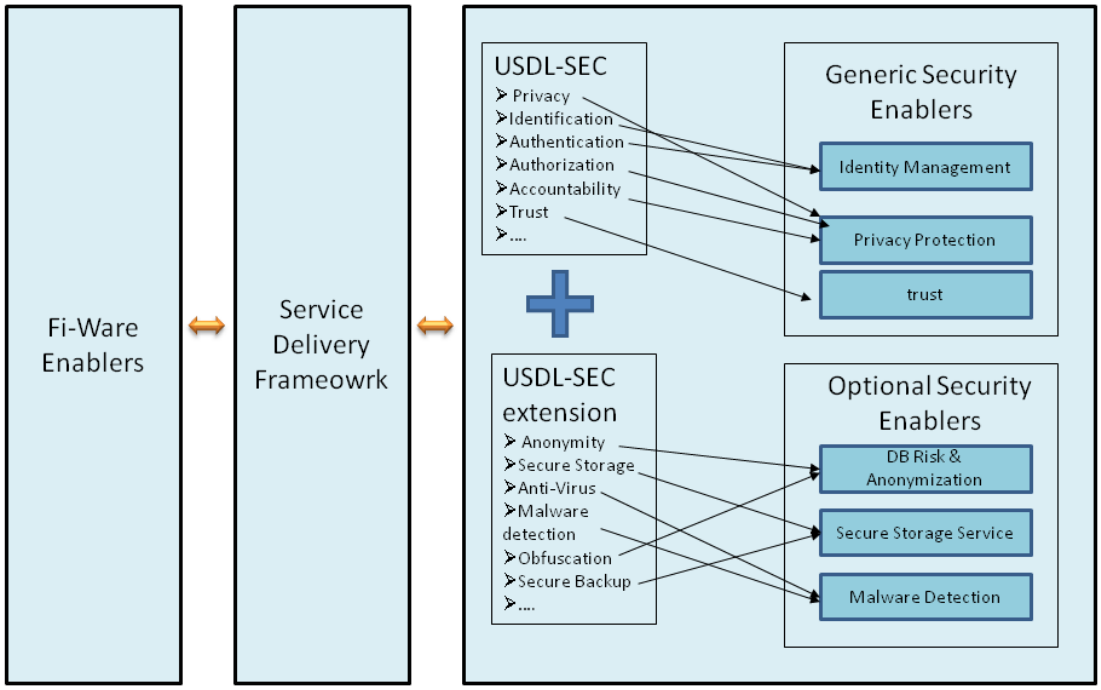


Figure 84 Optional Security Service Enabler

8.2.6.2 *GE description*

The optional security service enabler is used to customize the security service description within USDL-SEC when the security functionality is not covered by the specification. This asset targets directly the application domain usage. The goal is to make easily extendible the security service description for customized usage. This functionality will encourage all the developers to define and describe their own services through the USDL standard by adding new functionalities and new capabilities.

In Fi-Ware we want to overcome the limitations of the traditional service description languages that usually do not take into account the extensions wanted by the users. Most of the time the user has to add manually new elements and modify the delivery framework in order to take into account the new functionalities. Such approach is not feasible for any user. With this optional security services we will provide all the technical support to let users add and extend the USDL-SEC service description easily.

Some example of possible optional security services could be:

- **A database risk evaluation and anonymization service** used to check a shared database is not vulnerable to the re-identification of the non shared part of the database. The service can be used to support these DB administrators to evaluate the disclosure risk for all their types of data; by recommending the safest configurations using a smart bootstrapping system. The service will provide the user with a feedback on the re-identification risk when disclosing certain information and proposing safe combinations in order to help him during the information disclosure. Albeit privacy risk estimators have already been developed in some specific contexts (statistical databases), they have had limited impact, since they are often too specific for a given context, and do not provide the user with the necessary feedback to mitigate the risk. In addition, they can be computationally expensive on large datasets.

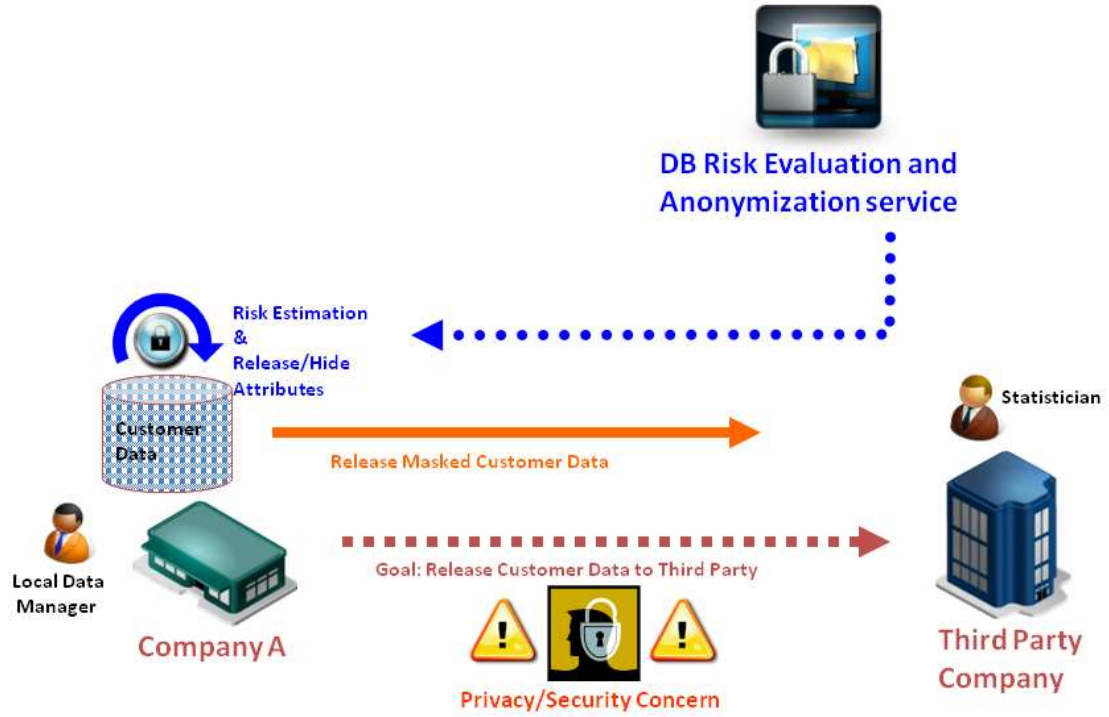


Figure 85 DB Risk Evaluation and Anonymization Service

- **Secure storage service** that offers the possibility to safely backup his data and delegates the access to parts of data to third party. Data leaks are resulting from the lack of additional information on data: sensitivity, access rights, lifetime, etc… The existence and the availability of these kinds of attributes are necessary to master the storage and the exchange of sensitive data. The concept consists in providing an secure storage service, manipulating self protected metadata only. This (optional) service can be used or not by other services, depending on the privacy level of implied data. The main objective of SSS is to provide a secure storage to sensitive / private data, privacy-oriented capacities, according to legislation.

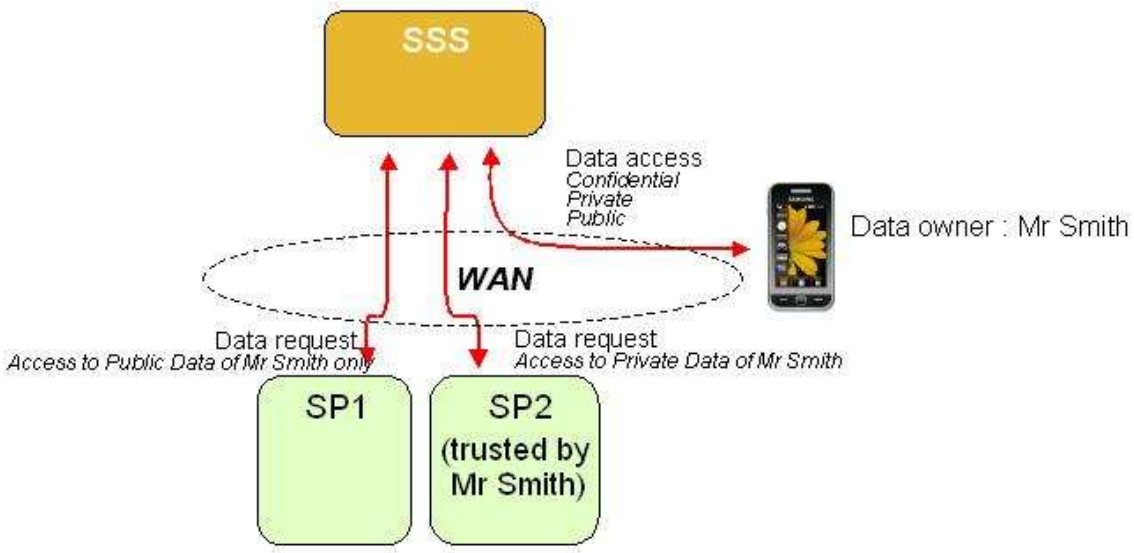


Figure 86 Secure Storage Service

- Morphus: **Malware detection service** that explores a data structure in order to check whether this dataset contains malware applications. Morphus is a generic malware detector based on graph signatures. Unlike the traditional notion of detection based on string pattern matching, graphs allow a behavioral analysis of programs, thus providing a much finer description of malwares.

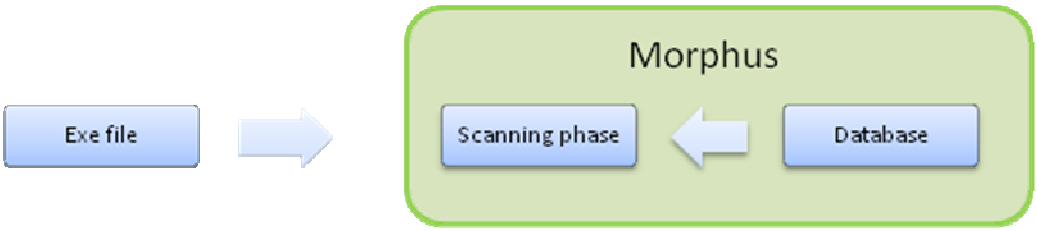


Figure 87 Morphus

8.2.6.3 *Critical product attributes*

- Make USDL-SEC service description extensible and flexible so that it can be used by anyone interested in making using of it (i.e.company, organization, individuals, ...)
- Provide a technical framework to deploy easily security services
- Optional security services can be deployed as plug-in applications.
- Break the description limits of the standard USDL specification

8.3 Question Marks

We list hereafter a number of questions that remain open. Further discussion on these questions will take place in the coming months.

8.3.1 Questions still under discussion

Forensics for evidence

The complexity problem in digital forensics is that acquired data are typically at the lowest and most raw format, which is often too difficult for humans to understand. It is not necessarily impossible, but often the skill required to do so is great, and it is not efficient to require every forensic analyst to be able to do so. To solve the complexity problem, tools are used to translate data through one or more layers of abstraction until it can be understood.

Similarly, the quantity problem in digital forensics is that the amount of data to analyze can be very large. It is inefficient to analyze every single piece of it. Data reduction techniques are used to solve this, by grouping data into one larger event or by removing known data. For example: identifying known network packets using Intrusion Detection System (IDS) signatures, unknown entries during log processing.

Proactive attack detection in a SOA context

It will very difficult to prevent attack in SOA context, if service providers are constantly changing.

Indeed, we must make, as a preliminary, a topological vulnerability analysis, using components and services stable enough to search their vulnerabilities and to identify the most likely paths of attack. The problem is that it is not possible, in this case, to identify such paths.

Business risk impact assessment

The difficulty to identify the attack paths in a volatile environment also can be applied to the business risk impact assessment. It is not possible to do it if the resources used by business process are always indeterminate.

Identification of the Information System

Identification of the Information System assets impacts the efficiency of the security monitoring GE and more specifically the efficiency of the vulnerability assessment capacity. System discovery is not expected to be a good solution because of false positives but also isolation mechanisms put in place (NAT, firewalls ...) which often hide numbers of assets. Moreover, this type of identification doesn't meet real time requirements. As a consequence, the support of an efficient configuration manager included in FIWARE (WP?) feeding a global Configuration Management Database (CMDB) is needed to achieve effective & efficient security monitoring through precise and real-time knowledge of assets.

Identity, Trust & Privacy management

The identity, trust, and privacy management component is concerned with authorization and authentication. This includes a (credential requirements) policy language to define with attributes (roles, identity, etc) and credentials are requested to grant access to resources. It further include a (data handling) policy language that defines how the requested data (attributes, credentials...) is handled and to whom it is passed on. Finally, it includes the means to release and verify such attributes and credentials. That raises three issues:

- The integration of the two policy languages into the access control system (e.g., XACML) of the FI-platform;
- The definition of the interfaces of the Generic Security Enabler;
- The integration of the different assets into components that realize the generic security enabler interfaces.

Accountability and Trust

The PPL privacy engine is designed to specify and enforce access and usage control rules on the collected data. But once the data collected and the privacy rules executed in theory how can we verify the accountability of the data controller? Accountability of the controller can be used to enhance the trust of the subjects that their data is not misused. One approach is to deploy an accountability mechanism for privacy policies which makes it possible to check a posteriori the logs of the system to check whether the actions performed over the data are compliant with the privacy regulation rules of the sticky policies. The fact that a controller agrees to implement a given compliance system can be a factor of trust for the subjects. Additionally, the level of compliance of each data controller could also be formalized as a trust metric used to describe the privacy reputation of a data controller.

Auditing

It must be possible to verify whether the system works as expected. (1) Logging of user activities in a standardized way is a key requirement. We should ensure that all the relevant data that is needed for a meaningful analysis is logged. (2) Logs may have to be exchanged across trust domains. We should provide mechanisms that support the analysis and correlation of logs that originated in different trust domains. Auditing also comprises the analysis of authorization and usage control policies based on meta-policies such as Separation of Duties (SoD) constraints.

New security/privacy threats related to Web2.0

How can we mitigate the threats that come from recent Web2.0 developments (e.g. “clickjacking”, “cookie jacking”, the threat that users of social networks such as Facebook trust their friends and hence easily accept malicious content from them, HTML5 geolocation API threat, etc).

Event interface definition

The event interface which will implement the communication between the Security enabler – User Application and Monitor systems would need to be defined asap.

Monitoring capabilities

Some of the monitor capabilities Context-based security and compliance will need to be developed, especially those referred to probe functionalities, may be fulfilled by Security monitoring enablers. Furthermore each of every monitoring capabilities addressed at WP8 level would have to be put back into perspective of overall monitoring capabilities offered by FI-WARE seen here as the FI PPP Core Platform.

Securing the deployment of new services

The definition and the deployment of the new optional security service require sometimes the composition with some generic security services (ex. Identity management, authentication, etc.). Then the service composition functionality should be supported by the FI-WARE service framework.

USDL-SEC specification

The current version of the USDL-SEC specification is a very early draft designed before the beginning of the Fi-Ware project. It does not reflect yet the security capabilities proposed in the generic security enablers exposed in WP8. The main task of the WP8 in the next months is to list these security capabilities and map them to a new version of the USDL-SEC specification in order to be able to publish correctly all the security services and make them available for any service deployed in the FI-WARE platform.

8.4 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)

- **Attack.** Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself
- **Business impact analysis.** An analysis of a business’s requirements, processes, and interdependencies used to characterize information system contingency requirements and priorities in the event of a significant disruption.
- **Countermeasures**. Action, device, procedure, technique or other measure that reduces the vulnerability of an information system.
- **Cyber attack**. An attack, via cyberspace, targeting an entity (industrial, financial, public...) and using cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment/infrastructure; or destroying the integrity of the data or stealing controlled information
- **Exploit.** A program or technique that takes advantage of vulnerability in software and that can be used for breaking security, or otherwise attacking a host over the network
- **Forensics for evidence.** The use of scientifically derived and proven methods toward the preservation,

collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.

- **Identity**. In the narrow sense, identity is the persistent identifier of users (user name), things or services by which other parties “remember” them and, hence, are able to store or retrieve specific information about them and are able to control their access to different resources. In the wider sense, identity also covers further attributes of users, things and services; e.g. for users, such information may include personal information such as context, group membership and profile.
- **Identity Management**. Identity Management (IDM) is a term related to how users, things and services are identified and authorized across computer networks. It covers issues such as how users, things and services are given an identity, the protection of that identity and the technologies supporting that protection such as network protocols, digital certificates, passwords and so on.
- **Impact**. The adverse effect resulting from a successful threat exercise of vulnerability. Can be described in terms of loss or degradation of any, or a combination of any, of the following three security goals: integrity, availability, and confidentiality. **Partial identity:** a partial identity is a set of attributes of a user. Thus, an identity is composed of all attributes of a user, a partial identity is a subset of a user's identity. Typically, a user is known to another party only as a partial identity. A partial identity can have a unique identifier. The latter is a strong identifier if it is allows for a strong authentication of the user (holder) of the partial identity, such a cryptographic "identification" protocol
- **Privacy**. Dictionary definitions of privacy refer to "the quality or state of being apart from company or observation, seclusion [...] freedom from unauthorized intrusion" (Merriam-Webster online [MerrWebPriv]). In the online world, we rely on a pragmatic definition of privacy, saying that privacy is the state of being free from certain privacy threats.
- **Privacy threats**. The fundamental privacy threats are: traceability (the digital traces left during transactions), linkability (profile accumulation based on the digital traces), loss of control (over personal data) and identity theft (impersonation).
- **Risk analysis**. The process of identifying security risks, determining their magnitude, and identifying areas needing safeguards. An analysis of an organization's information resources, its existing controls, and its remaining organizational and MIS vulnerabilities. It combines the loss potential for each resource or combination of resources with an estimated rate of occurrence to establish a potential level of damage
- **Security monitoring**. Usage of tools to prevent and detect compliance defaults, security events and malicious actions taken by subjects suspected of misusing the information system.
- **S&D:** Security and Dependability
- **Threat.** An event, process, activity being perpetuated by one or more threat agents, which, when realized, has an adverse effect on organization assets, resulting in losses (service delays or denials, disclosure of sensitive information, undesired patch of programs or data, reputation...)
- **USDL and USDL-Sec:** The Unified Service Description Language (USDL) is a platform-neutral language for describing services. The security extension of this language is going to be developed FI-WARE project.
- **Vulnerability.** A weakness or finding that is non-compliant, non-adherence to a requirement, a specification or a standard, or unprotected area of an otherwise secure system, which leaves the system open to potential attack or other problem.
- **WS-SecurityPolicy:** It is an extension to SOAP to apply security to web services. It is a member of the WS-* family of web service specifications and was published by OASIS.
- **The protocol** specifies how integrity and confidentiality can be enforced on messages and allows the communication of various security token formats, such as SAML, Kerberos, and X.509. Its main focus is the use of XML Signature and XML Encryption to provide end-to-end security.

8.5 References

[biccam]	P. Bichsel and J. Camenisch. Mixing Identities with Ease. IFIP Working Conference on Policies & Research in Identity Management (IDMAN '10), Oslo, Norway, November 18-19, 2010.
[EuPriv95]	EU Directive 95/46/EC - The Data Protection Directive
[GaaBook08]	Dr. Silke Holtmanns, Valtteri Niemi, Philip Ginzboorg, Pekka Laitinen, Prof. N. Asokan. (2008) "Cellular Authentication for Mobile and Internet Services". Wiley, ISBN: 978-0-470-72317-3 October 2008.
[idemixspec]	IBM Research – Zurich, Specification of the Identity Mixer Cryptographic Library, Version 2.3.3 http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/IdentityMixer_ProtocolSpecification_2-3-3.pdf
[idemixlib]	http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/index.html
[OpenId]	OpenID. http://openid.net/
[Ppl10]	Slim Trabelsi and Akram Njeh and Laurent Bussard and Gregory Neven,"The PPL Engine: A Symmetric Architecture for Privacy Policy Handling", W3C Workshop on Privacy and data usage control 04/05 October 2010, Cambridge (MA), USA.
[Saml]	Security Assertion Markup Language (SAML) v2.0. http://www.oasis-open.org/specs/index.php#samlv2.0
[Xacml]	OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/xacml/
[IoS]	USDL - Internet of Services http://www.internet-of-services.com/