FI-WARE Product Vision front page

EUROPEAN COMMISSION

Community Research

SEVENTH FRAMEWORK PROGRAMME

Large-scale Integrated Project (IP)

fi-ware

D2.2: FI-WARE High-level Description.

Project acronym: FI-WARE

Project full title: Future Internet Core Platform

Contract No.: 285248

Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform

Project Document Number: ICT-2011-FI-285248-WP2-D2.2b

Project Document Date: 15 Nov. 2011

Deliverable Type and Security: Public

Author: FI-WARE Consortium

Contributors: FI-WARE Consortium.

Abstract: This deliverable provides a high-level description of FI-WARE which can help to understand its functional scope and approach towards materialization until a first release of the FI-WARE Architecture is officially released.

Keyword list: FI-WARE, PPP, Future Internet Core Platform/Technology Foundation, Cloud, Service Delivery, Future Networks, Internet of Things, Internet of Services, Open APIs

Contents

Articles

FI-WARE Product Vision front page	1
FI-WARE Product Vision	2
Overall FI-WARE Vision	3
FI-WARE Cloud Hosting	13
FI-WARE Data/Context Management	43
FI-WARE Internet of Things (IoT) Services Enablement	94
FI-WARE Applications/Services Ecosystem and Delivery Framework	117
FI-WARE Security	161
FI-WARE Interface to Networks and Devices (I2ND)	186
Materializing Cloud Hosting in FI-WARE	217
Crowbar	224
XCAT	225
OpenStack Nova	226
System Pools	227
ISAAC	228
RESERVOIR	229
Trusted Compute Pools	230
Open Cloud Computing Interface (OCCI)	231
OpenStack Glance	232
OpenStack Quantum	233
Open vSwitch	234
Claudia	235
PaaS Management Platform	237
OpenStack Swift	239
Nada Management Framework	240
OpenStack Keystone	241
Materializing Data/Context Management in FI-WARE	242
Context Awareness Platform (CAP)	248
AMIT	249
Hadoop	251
MongoDB	251
SAMSON Platform	252
Codoan	253
White Diamond	254

Meta-data Processor 255
Location server 256
Location Provider 257
QueryBroker 258
Semantic Annotator (SAnr) 259
NeOn Toolkit 260
Sesame 262
OWLIM 263
Materializing Internet of Things (IoT) Services Enablement in FI-WARE 264
Cumulocity 274
FossTrak 277
M2MPlanet/Pangoo 278
IDAS 279
ISIS 281
IoT-A 282
Atos SOL 283
Sensei 284
Linked Data Platform and Gateway 286
Materializing Applications/Services Ecosystem and Delivery Framework in FI-WARE 288
USDL 3.0 Specifications M5 295
Apache Chemistry Client and Server 296
JENA SDB/TDB RDF Store 297
USDL Service Marketplace OS 298
PREMIUM-Services Pricing Strategies Simulator 300
Ericsson eStore 301
Revenue Sharing System 302
EricssonCommonCompositionCore 303
WebConnectivityEnabler 305
Wirecloud Mashup Platform 307
Mashup Factory 310
LighSemantic-enabled Design Time Semi-automatic Service Composition 311
WSO2-Mediation-Layer-Extension 312
Light-Semantics-enable Process and Data Mediation Composer 313
SLA Model 314
Materializing Security in FI-WARE 315
IoT Internet protocols fuzzing framework 320
FI-Ware Vulnerability assessment 321
Ontology handler 321

Vulnerabilities OVAL scanner 322
NVD 323
Attack trace engine 324
Service-Level-SIEM 325
CVSS 326
Visualization Framework 328
Fragmento 329
Compliance Governance Dashboard 331
CRLT Compliance Request Language Tools 333
USDL Language 334
S&D Run-Time Framework 336
Stork 338
Identity Management 341
White Label IdP 342
Access Control 345
Privacy Enabler 347
Idemix - Privacy-Preserving Credential Library 348
Idemix - Credential-based Authentication Engine 350
Accountable privacy policies 352
PrimeLife Policy Engine: PPL 353
Database Anonymization Optional Asset 356
Secure Storage Service (SSS) Optional Asset (Thales) 359
Morphus Optional Asset (INRIA) 362
Materializing the Interface to Networks and Devices (I2ND) in FI-WARE 363
WEB Runtime Engine 367
Meego 368
Web API 369
Reinforcement Learning 371
Network Organizer 372
ZigBee GAL 373
NOX based OpenFlow controller 374
Network Element Virtualizer 374
DRAGON 375
OpenEPC Platform 376
Device Connection Platform 378
API mediation 381
Fast Telecom Service helper 382
Voicemail enabler 383

Intelligent Network Identity Management 384
Network-centric Resource Management 385
Seamless Network Connectivity 386

References

Article Sources and Contributors 388
Image Sources, Licenses and Contributors 391

FI-WARE Product Vision

Introduction

The FI-WARE Product Vision provides a high-level description of the FI-WARE Platform. FI-WARE aims to provide a framework for development of smart applications in the Future Internet.

FI-WARE is being developed as part of the Future Internet Public Private Partnership (FI-PPP) program launched by the European Commission in collaboration with the ICT Industry. However, it has a global ambition. More information about the FI-PPP program can be found at:

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

More information about FI-WARE can be found at:

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

FI-WARE Chapters

You should start reading the Overall FI-WARE Vision. It will give you an overview of some basic concepts and principles that are general to FI-WARE (Generic Enablers, FI-WARE Instances, etc).

The Reference Architecture of the FI-WARE platform is structured along a number of technical chapters, namely:

- Cloud Hosting
- Data/Context Management
- Internet of Things (IoT) Services Enablement
- Applications/Services Ecosystem and Delivery Framework
- Security
- Interface to Networks and Devices (I2ND)

You can get a description of each of these chapters by clicking on the chapter name in the list above. Each chapter description is structured so that a general overview of the chapter is provided first, where main Generic Enablers in the chapter are identified and briefly described. Afterwards, a more detailed description of each of these Generic Enablers is provided. A precise list of terms and definition is provided for each chapter 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.

The FI-WARE project will draw upon the wealth of results already achieved through earlier Research projects, not only within the EU FP but also at national- or corporate-funded levels, aiming to leverage them further through a systematic integration with a complete system perspective. A description of what are the products (assets) that will be considered as baseline for the development of a first reference implementation of each Generic Enablers in FI-WARE can be found in Materializing the FI-WARE Vision. They will evolve and will be integrated together following an Agile development process you can monitor following the definition and evolution of backlog entries associated to the chapter and/or each of the Generic Enablers. A description of projects whose results may be be considered for further integration is provided in those cases where implementation of the Generic Enabler is not planned for the current FI-WARE major release.

Overall FI-WARE Vision

Background and Motivations

Changing ICT Landscapes and Trends

Over the last couple of years, a number of technology trends are recognized as significant new directions in the ICT landscape, which will usher in the era of the Future Internet in Europe. A first major trend is the on-going industrialization of IT in the form of cloud computing and open service delivery platforms. The on-demand, pay-per-use nature of these provisioning models is expected to fundamentally change how IT infrastructures are provisioned, essentially enabling them to be delivered as a service. Secondly, new wireless networking technologies such as LTE (4G) and the deployment of Fibre to The Home (FTTH) will offer increased network capacity and bandwidth to customers, thereby paving the way for new (real-time) applications in mobile data networks as well as for stationary scenarios. Furthermore, the virtualization of classical carrier networks is also on-going and starting to find new areas of application. Thirdly, the Internet of Things is safely taking hold, with the vision of ubiquitously connecting intelligent devices and sensors and offering new possibilities for contextual and environmental information sensing, processing and analysis, thereby opening the door to new automation and control applications and services in various sectors. Lastly, the maturing Internet of Services is accelerating the creation of complex value networks of service providers, consumers, and intermediaries bringing businesses and end customers innovative applications better tailored to their needs. These networks increasingly span various different players that historically have worked largely separated from each other thereby leading to more agile and dynamic business relationships and environments never seen before.

The Market Stakeholder Perspectives

Brought together, the above trends are the core drivers towards the emergence of a new era in the evolution of the Internet, which we may refer to as Future Internet. It will carry the potential for realizing new business opportunities for established and emerging application and service providers in areas such as telecommunications, healthcare, media and e-government. This Future Internet will address some key demands and expectations from the various market stakeholders which cannot yet be satisfactorily met. Thus, regarding Application and Service consumers:

- **End customers** want to gain access and easily consume applications that can effectively assist them in daily life situations. Some of the underlying problems involved are the management of the ever-growing data and information (e.g. from their sensor-enabled environments) and the seamless access anywhere, anytime and from any device. They also ask for improved means for communication and collaboration within their social networks, families, neighbourhoods in real-time and while being mobile, meeting security and privacy requirements. Overall, these capabilities would transform communities, homes and cities into safer and better places to live and leverage the Internet as an additional societal utility.
- **Enterprises and organizations** on the other hand, wish to get closer to their customers in order to deliver an even more compelling user experience and better service. For this reason, they would like to exploit contextual user data which may lead to a more personalized interaction experience and service offering, and would like to realize a stronger participation of users in all phases of product and service lifecycles, thereby bringing the lessons of the Web 2.0 phenomena into the services space. In order to develop and operate their services, new methods, technologies and tools are needed to speed up the time to market, to establish value added services which may be better configured in partnership with others and to simplify access to relevant resources and capabilities, e.g., from the Internet of Things. Additional requirements on business services include reduced complexity of ICT provisioning, scaling, global availability and meeting security requirements from customers and legal authorities. An appropriate Future Internet platform would greatly contribute to meeting these demands from business

customers.

Application/service developers and providers, on the other hand, are challenged to build smart applications that cover the above mentioned needs from consumers. From a development perspective, such applications and services should be built on top of powerful but easy to use APIs, be standards based and offer flexible deployment and provisioning means (e.g., many devices, multi-tenant architectures, global scalability, and on-demand provisioning) and management frameworks (e.g. in the Internet of Things). Additionally, they should exploit economies of scale and protect investments in the long run. Finally, the ability to combine applications from different sources necessitates innovative revenue sharing models across partners and potentially also customers (e.g. crowd-sourcing) which have to be adapted dynamically as market conditions change.

Towards Strengthening Innovation-enabling Capabilities

The previous observations independently hold across vertical sectors such as health, transportation and logistics, energy and urban management. However, in practice, many solutions in those areas are today realized as custom made applications which are developed multiple times for particular purposes with proprietary interfaces. This severely hinders the growth of economies of scale for application and service developers, and limits the size of the addressable markets and ICT investments that can be made towards new products and services in vertical applications sectors. As most revenues in the ICT sector in 2020 will be generated by products and services which have not yet been developed, it is now commonly agreed by public and private thought leaders that investments into the innovation-enabling capabilities are crucial for success in the global market competition.

Economical, Societal, and Political Benefits brought by a FI Core Platform

From an **economic perspective** it can be observed that many companies in the traditional ICT sector face difficulties concerning the transformation of their own business models into new areas, tackling commoditization and marginalization threats. To address this difficulty, a framework is needed where new business models can be explored and validated effectively. Such a framework could help to cultivate an ecosystem comprised of agile and innovative service providers, which in turn consume services provided by the traditional ICT players.

Considering the **societal dimension**, the availability of a platform whereby stake holders across different sectors (e.g., healthcare, logistics, energy management, sustainability, transport etc.) can cooperate will accelerate the development of new innovative services for the European society within and across various sectors. Vertical and horizontal innovations in these areas will contribute to solving major societal challenges (e.g., Grand Societal Challenges identified by the EU Commission and EU Member States).

Lastly, on the **political dimension**, legal and legislative barriers presently hinder the efficient cross-border establishment of new innovative solutions due to complex or incompatible ICT policies in different countries and regions. The identification of legal and regulative aspects that could be potential barriers for innovation in the Future Internet should thereby be investigated and appropriate mitigation actions should be identified and brought to the attention of policy makers. A FI Core Platform may serve the purpose of revealing such barriers.

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. Building 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 a very limited set of Usage Areas. While not all elements that are suitable to be considered as Generic Enablers will be developed in the FI-WARE project (this would be a goal that the FI-WARE project alone, simply cannot afford), the intention is that all elements developed within FI-WARE can widely be accepted as Generic Enablers, per definition above.

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 specification of GEs, relations among them and properties of both.

The Core Platform to be provided by the FI-WARE project is based on GEs linked to the following main **FI-WARE Technical Chapters**:

- Cloud Hosting – the fundamental layer which provides the computation, storage and network resources, upon which services are provisioned and managed.
- Data/Context Management – the facilities for effective accessing, processing, and analyzing massive streams of data, and semantically classifying them into valuable knowledge.
- Applications/Services Ecosystem and Delivery Framework – the infrastructure to create, publish, manage and consume FI services across their life cycle, addressing all technical and business aspects.
- Internet of Things (IoT) Services Enablement – the bridge whereby FI services interface and leverage the ubiquity of heterogeneous, resource-constrained devices in the Internet of Things.
- Interface to Networks and Devices (I2ND) – 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 Application/Service developers. Therefore, FI-WARE will establish tools and processes that will help to closely collaborate with concrete Use Case projects dealing with development of concrete Application/Services. Discussions about requirements will require continuous interaction between the FI-WARE project and partner Use Case projects. However, FI-WARE development needs to be driven by requirements extrapolated for any future

Application/Service. Requirements that go beyond those of concrete Use Case projects will be brought by partners of the FI-WARE consortia (based on input of their respective Businesses Units). A FI-WARE product backlog will be generated from both sources and will be continuously updated, driving development of FI-WARE following Agile principles.

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.

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 missions listed previously.
- Design extension mechanisms so as to enable to support for yet unforeseen Usage Areas not being addressed initially. 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 standardization bodies in order to: a) keep the project up-to-date with respect to the discussions in the standardisation bodies; b) 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 Applications/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.

FI-WARE Generic Enabler Open Specifications, Compliant Platform Products, Instances

Specifications of APIs (Application Programming Interfaces) and Interoperable Protocols supported by FI-WARE Generic Enablers (GE) will be public and Royalty-free. 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. GE Open Specifications will typically include, but not necessarily will be limited to, information such as:

- Description of the scope, behaviour and intended use of the GE
- Terminology, definitions and abbreviations to clarify the meanings of the specification
- 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

The FI-WARE consortium intends to deliver a reference implementation for each of the Generic Enablers defined in the FI-WARE Architecture. Some components of these reference implementations may be closed source while others may be open source. The concrete open source license selected by the owning partners who work together in the implementation of a given component will be agreed by them, taking into account the Access Rights obligations and avoiding any impact on other Project partners they don't desire.

Note that not all components in a compliant implementation of a GE need to interoperate with components linked to implementations of another GE, nor provide APIs (Application Programming Interfaces) to Application Developers. While implementations of Generic Enablers developed in compliance with FI-WARE GE Open Specifications should be replaceable, components linked to a particular implementation of a Generic Enabler may not.

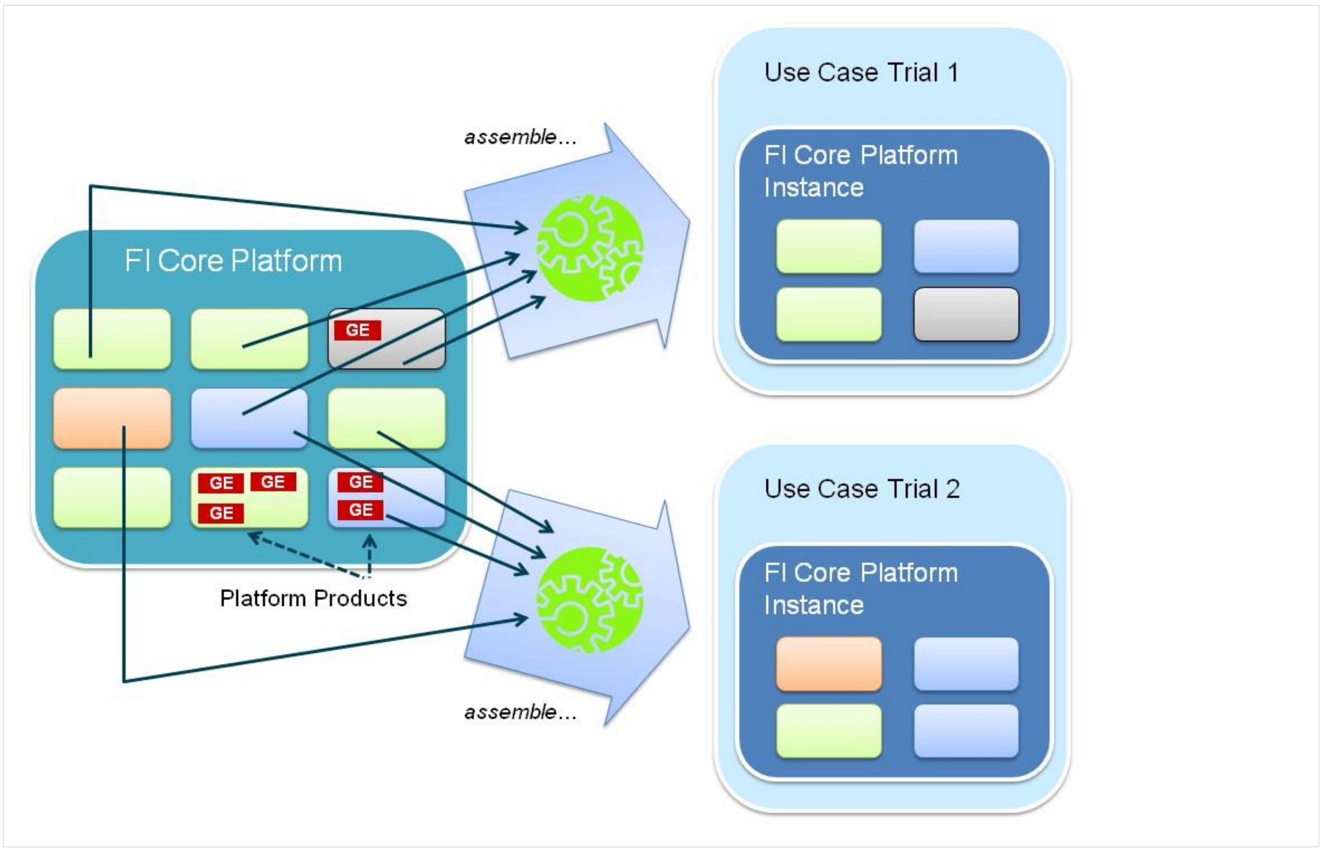
The FI-WARE project will draw upon the wealth of results already achieved through earlier research projects, not only within the EU FP7 but also at national- or corporate-funded levels, aiming to leverage them further through a systematic integration with a complete system perspective. In the FI-WARE project, there is an equal focus on advancing 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

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.

Note that the open nature of GE specifications will allow to replace a GE implementation developed in FI-WARE within a particular FI-WARE Instance.

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.



FI-WARE Instances

The FI-WARE Testbed and Open Innovation Lab

The FI-WARE project will generate a FI-WARE Instance, hereunder referred to as **FI-WARE Testbed**, which will allow partner Use Case projects (including a number of Use Case projects that are part of the European FI PPP initiative) to run and test Future Internet Applications based on FI-WARE Generic Enablers. This FI-WARE Testbed will be available shortly after delivery of the first FI-WARE major release. FI-WARE Instances linked to trials or commercial services (exploitation phase) are, in turn, referred to as "**FI-WARE Instances in production**".

The FI-WARE Testbed is aimed to be complete, in the sense that it will comprise reference implementations of all Generic Enablers defined in the FI-WARE Architecture. The testbed will not necessarily be centralised, but will be under central control and be accessible from a dedicated website. The FI-WARE partners will provide support to Use Case projects for the deployment of applications (e.g., the conceptual prototypes) on top of the FI-WARE testbed. Tests run by the partner Use Case projects, coordinated with tests defined by the FI-WARE project, will help to validate Generic Enabler Open Specifications, the reference implementations of FI-WARE Generic Enablers developed within the FI-WARE project, as well as the conceptual prototypes developed by Use Case projects.

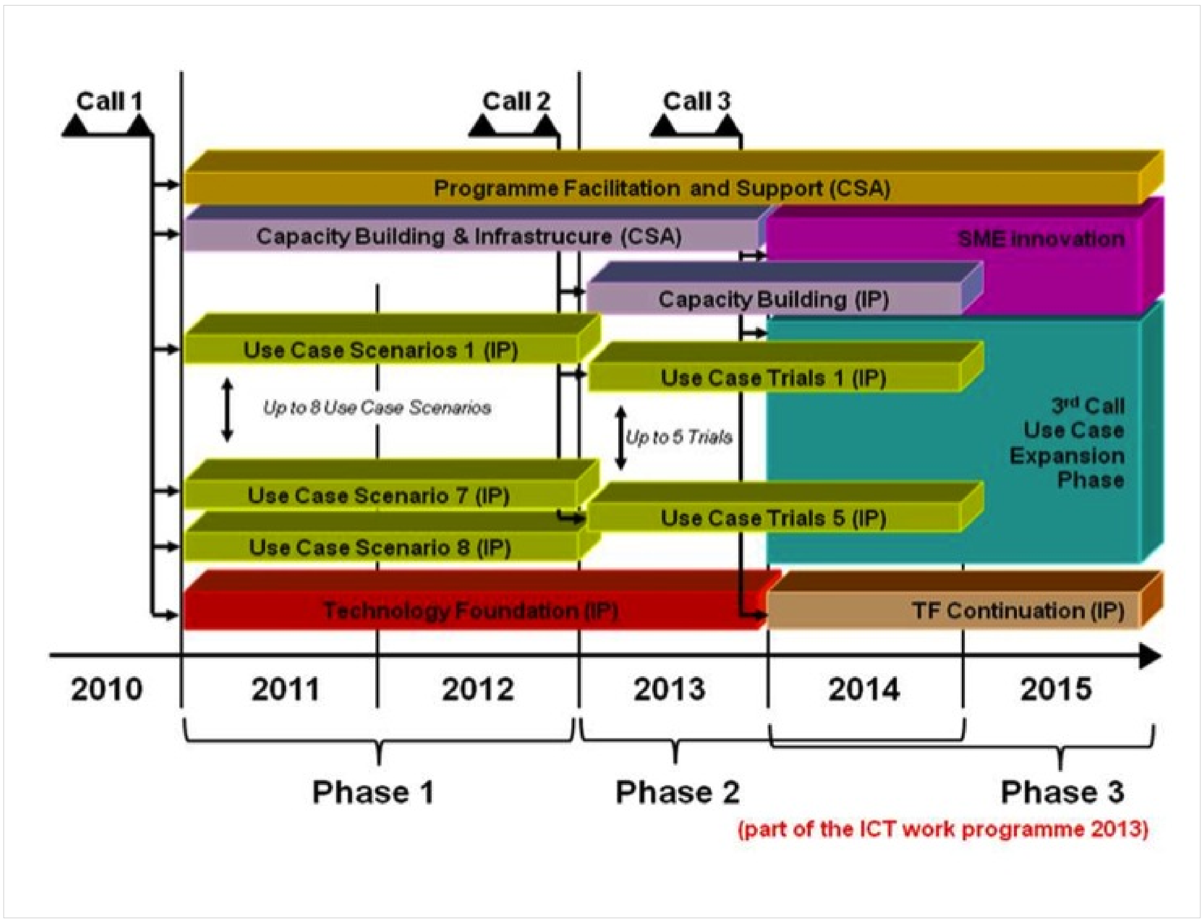
In order to pave the way for a successful exploitation and sustainability, the FI-WARE project will work on setting up an **Open Innovation Lab** around the FI-WARE testbed afther the second release of FI-WARE. This Open Innovation Lab will support community involvement beyond the initial partner Use Case projects, offering a space where future innovations on top of the generic enablers provided by FI-WARE can be nurtured. Availability of the FI-WARE testbed per se does not guarantee innovation as such, therefore the FI-WARE Open Innovation Lab will comprise all what is needed to stimulate awareness among target application providers and users so that they feel attracted to participate and build a community. It will also bring tools helping members of the community to share their experiences, needs, etc.

FI-WARE in the context of the European Future Internet Public Private Partnership (FI-PPP) Program

The FI-WARE project will design, develop and implement the so-called Core Platform within the European Future Internet Public Private Partnership (FI-PPP) Program defined under the ICT FP7 Work Programme. More information about the 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 following figure illustrates the basic FI-PPP approach, where several partner Use Case projects (eight in Phase 1 of the program) will cooperate with the FI-WARE project to provide requirements about Generic Enablers which are distinguished from Domain-specific Common Enablers (see previous sections). The identified requirements from the eight Use Case projects provide part of the requirements, which FI-WARE has to fulfil. Other requirements come from other partner Use Case projects that may arise during lifetime of the FI-WARE project or from the Business Units of the partners in the FI-WARE Consortia.


Call 1
Call 2
Call 3
Programme Facilitation and Support (CSA)
Capacity Building & Infrastrucure (CSA)
SME innovation
Capacity Building (IP)
Use Case Scenarios 1 (IP)
Up to 8 Use Case Scenarios
Use Case Scenario 7 (IP)
Use Case Scenario 8 (IP)
Use Case Trials 1 (IP)
Up to 5 Trials
Use Case Trials 5 (IP)
3rd Call Use Case Expansion Phase
Technology Foundation (IP)
TF Continuation (IP)
2010
2011
2012
2013
2014
2015
Phase 1
Phase 2
Phase 3
(part of the ICT work programme 2013)

FI-PPP Program

A first release of the reference implementation of GEs in FI-WARE will be provided before the end of phase 1 in the context of the European Future Internet Public Private Partnership (FI-PPP) programme and can be integrated to setup FI-WARE Instances serving Usage Trial projects in phase 2 of that programme.

Business Ecosystem

The following table describe the different roles in the overall value chain envisioned 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 platform 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.
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 will enable portability of FI applications and/or services across different FI-WARE Instances.

While there are clear boundaries among these roles, a company/organization may take one or more of the roles defined above. Indeed, we foreseen it will be very common that some companies/organizations play one of these roles as its core role but will try to gain differentiation in the market or try to capture additional revenue streams by playing any of the other complementary roles. Thus, scenarios like the followings may apply:

- A FI-WARE Application/Service Provider may also play the role of FI-WARE Instance Provider, deploying FI-WARE Instances tailored to run the Applications and Services they have developed (i.e., following a sort of ASP model). Conversely, a FI-WARE Instance Provider may decide to develop some Application/Services targeted to end customers on top of the same FI-WARE Instance they offer to third-party for development of their respective Application/Services.
- A FI-WARE Application/Service Provider may also decide to provide their own implementation of some GEs that will bring a differential value when bundled together with the final Application and Service they provide. Therefore, they may also play the role of GE provider.
- A FI-WARE Instance Provider may decide to be the Provider (implementer) of some of the GEs in the FI-WARE Instance it operates. This may be because of costs (no need to buy licenses or pay per use of a particular FI-WARE GE compliant product) or because it believes it may gain some differentiation by means of implementing part of the FI-WARE GEs against other FI-WARE Instance Providers.
- Considering all the previous points, a company may decide to play all of the three roles.

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. The following categorisation summarises the relevant stakeholders, the roles they are expected to play and, consequently, the impact FI-WARE may induce:

Established Telecom Industry

- Telecom Service Providers: They will typically play the role of FI-WARE Instance Providers as their core role, relying on FI-WARE technologies to develop new open and innovative business models leveraging the assets they already have, i.e. exploring possibilities to "open up" their existing networks or the data they manage or connecting third Application/Services Providers with their large customer base. In general, trying to create a compelling ecosystems for Application/Services Providers that leverages on revenue share models. They, of course, will need to understand the technical hurdles, which need to be overcome. They can also play a relevant role as FI-WARE GE Provider and accelerating the development of standards based on FI-WARE results. They also may play the role of FI-WARE Application/Service Providers, developing 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. Also to leverage their position as FI-WARE Instance Provider or simply extend its portfolio, thus being able to keep growing in the Digital world.

- Network equipment manufacturers (core and access networks): Develop and provide technical solutions to support Telecom Service Providers in their role as FI-WARE Instance Providers. They may generate product/platforms covering a number of FI-WARE GEs and provide services on which Telecom Service Providers may rely to implement their role as FI-WARE Instance Providers or FI-WARE Application/Service Prroviders. 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 or to enrich the Applications/Services ecosystem a given Telecom Service Provider wishes to build around the FI-WARE Instance it operates.
- 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. Some of these new capabilities may require they implement some FI-WARE GEs, so that they are more suitable to run on their devices. Based on open standardized interfaces, economies of scale and affordable/interchangeable solutions for customers can be achieved.

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.
- IT Providers: Many of them are evolving in a line similar to Telco Service Providers, therefore playing any of the three defined roles.
- IT Solution integrators: They will typically play the role of FI-WARE Application/Service Providers, adapting to the new challenges of developing and integrating new converged Telecom/IT solutions. They sometimes may decide to play the role of FI-WARE Instance Providers, offering the outsourcing of all ingredients that are part of the solution.

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.

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.

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.

Summary of key concepts introduced

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

- **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 published for that GE.
- **FI-WARE GE Open Specifications**: 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
- **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 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 European FI PPP initiative) 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.

FI-WARE Cloud Hosting

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.

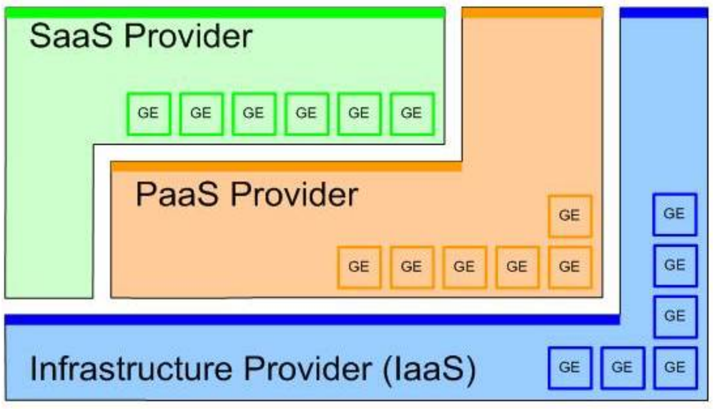
The FI-WARE project focuses a great part of its efforts on making sure that these standards materialise and in facilitating their adoption by providing open specifications and reference implementations. This standards-based and open approach will cover the fundamental technologies on which the cloud 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 [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.



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:

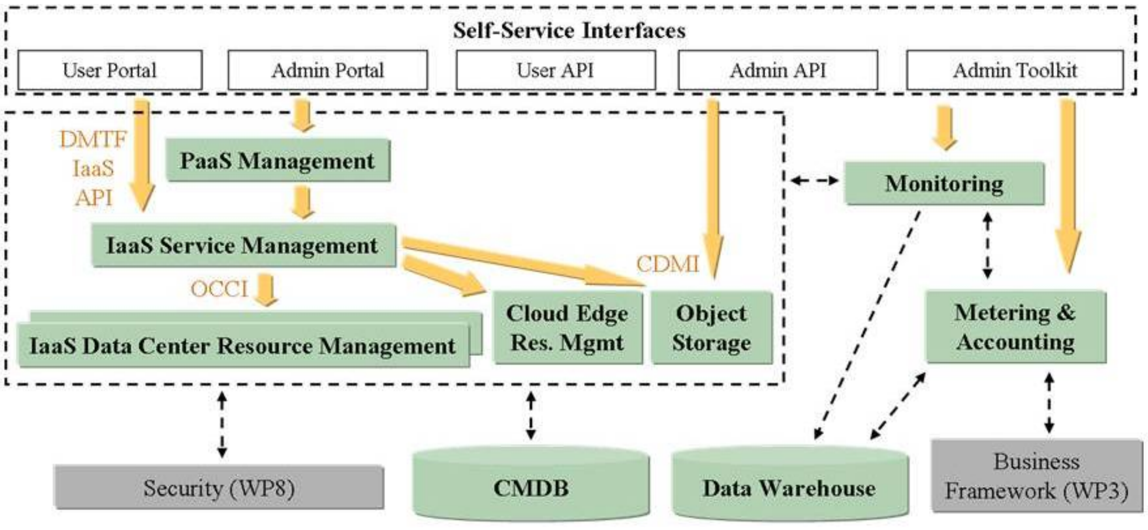
Building upon existing virtualization technologies, FI-WARE will deliver a next generation Cloud Stack that will be open, scalable, resilient, standardised, and secure, and will enable Future Internet applications by providing service-driven IaaS and PaaS functionalities and extending the reach of the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

The following figure 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.



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.

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.

Generic Enablers

IaaS DataCenter Resource Management

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

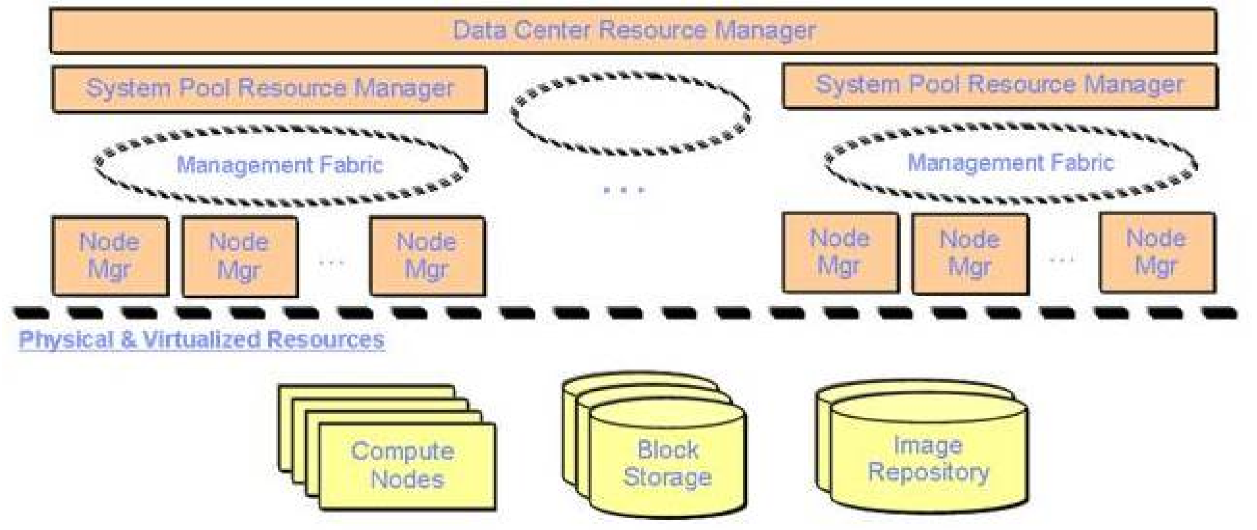
For a **cloud hosting provider**, the following capabilities are provided:

- 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)

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.



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].

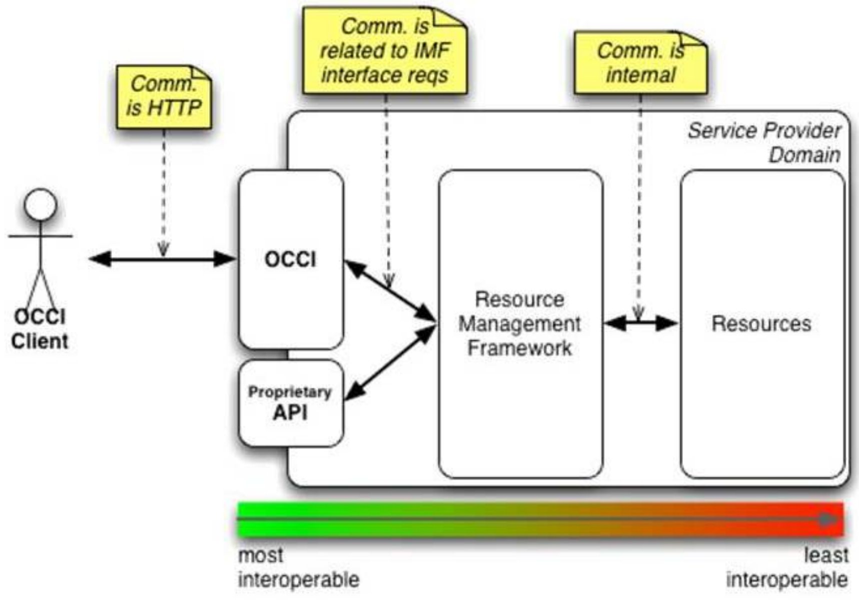
The main design foci of OCCI are:

- 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.



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 (1) and a number of supporting tools(2). It is has been recommended by the UK G-Cloud initiative(3), is currently in the process of consideration by NIST(4) in the US and also supported by the SIENA(5) and EGI(6) 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(7) and SLA capabilities(8).

1. - http://www.occi-wg.org/community/implementations
2. - http://www.occi-wg.org/community/tools
3. - http://occi-wg.org/2011/02/21/occi-and-the-uk-government-cloud/
4. - http://www.nist.gov/itl/cloud/sajacc.cfm
5. - http://www.sienainitiative.eu/
6. - http://www.egi.eu
7. - http://www.iolanes.eu

8. - http://en.wikipedia.org/wiki/D-Grid

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.

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 p1 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.

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

IaaS Service Management

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).

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

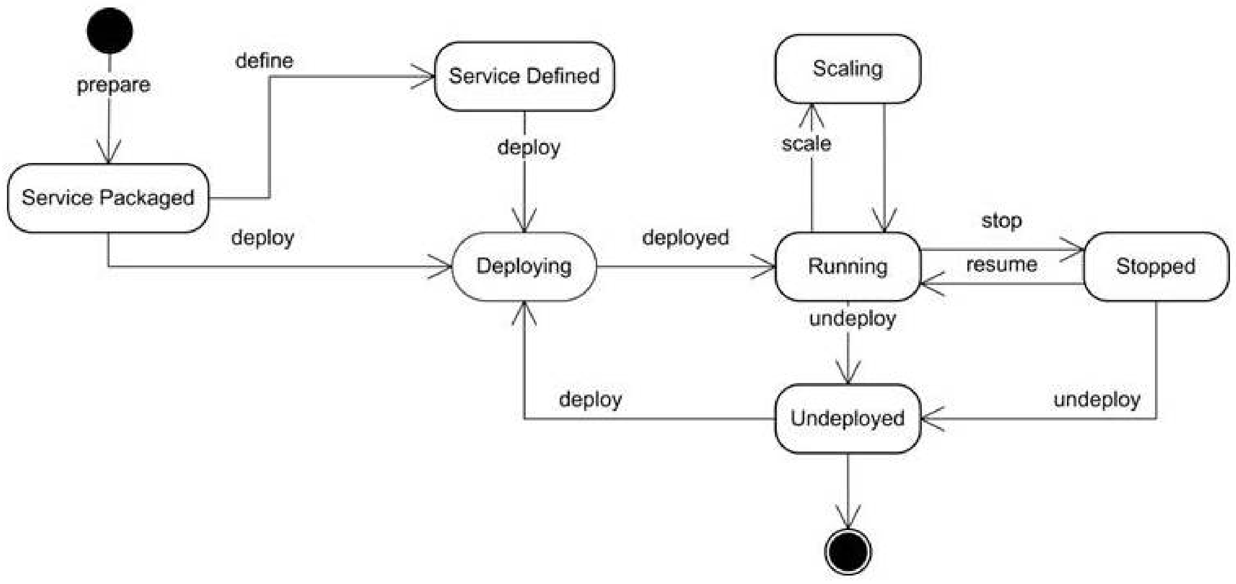
As FI-WARE works with different sites and with a variety of services, it is required that virtual machines will be portable between sites. It is required to guarantee interoperability in service and virtual machine definitions between the sites. In a cloud environment, one key element of the interaction 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.



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 driver integration	The API implementation will include drivers to handling access to the IaaS 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

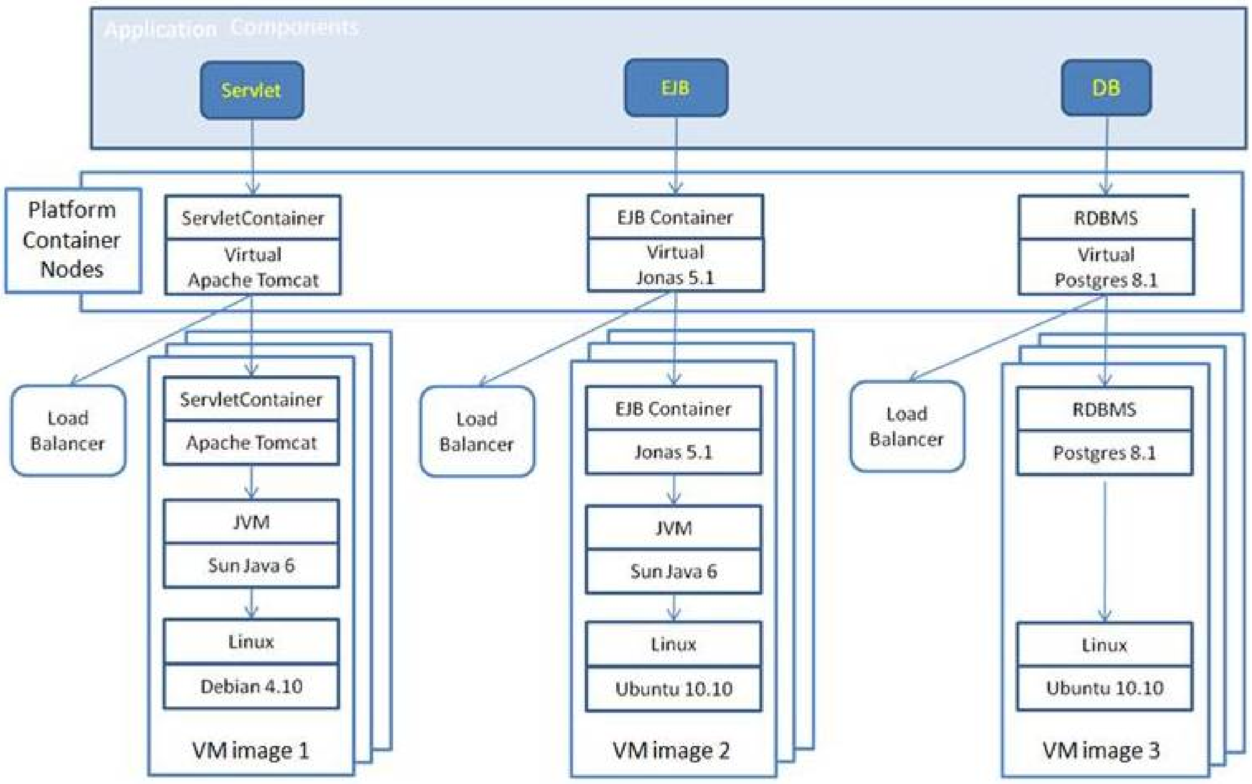
Critical product attributes

- 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.

PaaS Management

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.



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..

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.

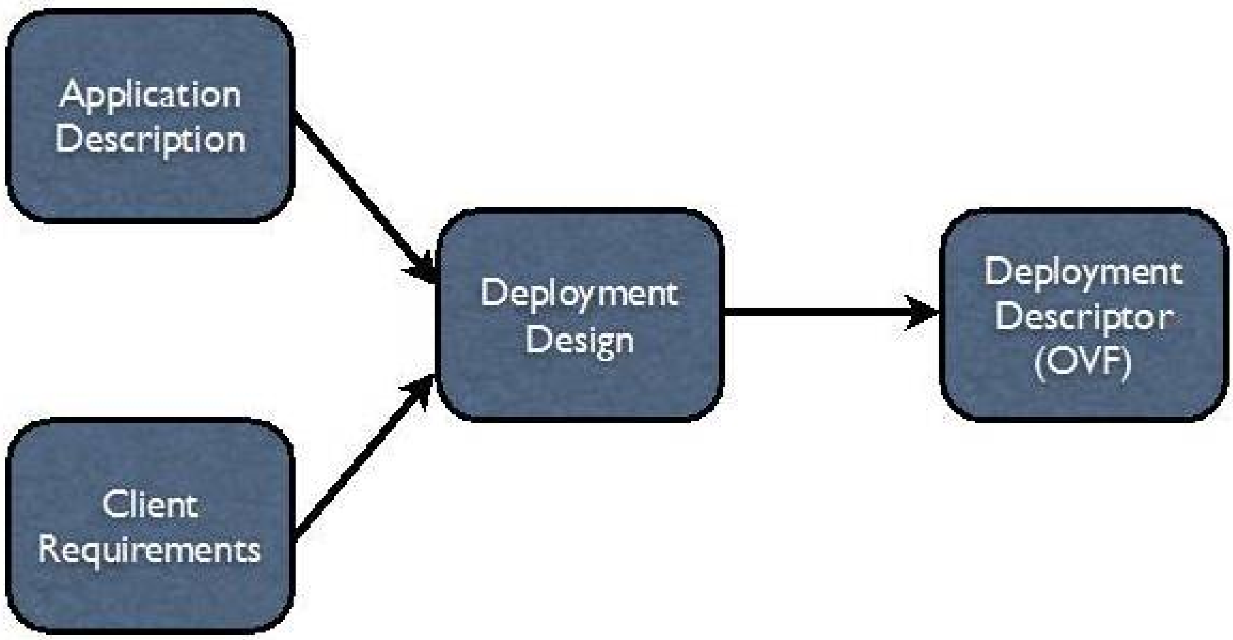
The input of this process will be the Application Description (AD). Based on the requirements imposed on the different Application Components (ACs) the PaaS Manager GE will generate an Application Deployment Descriptor (ADD) which will comprise an initial IaaS Service Manifest or set of changes to existing 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.



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.

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.

Object Storage

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).

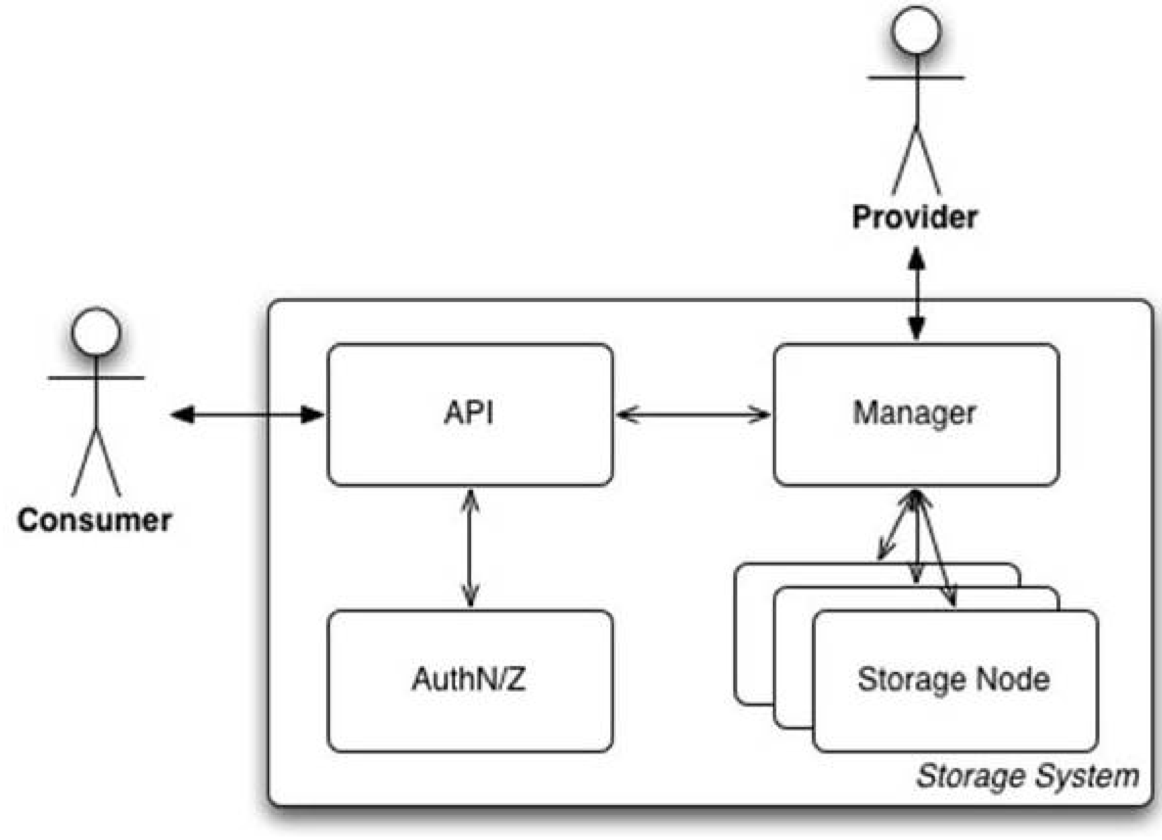
- FCIP usage: The CHP can take two roles: one as consumer of the service and another as its manager. From a consumer perspective, the FCIP will use this system in order to store certain types of data. A good example of this is in the storage of monitoring, reporting and auditing data. That data could 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.

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:



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

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

IaaS Cloud-edge Resource Management

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.

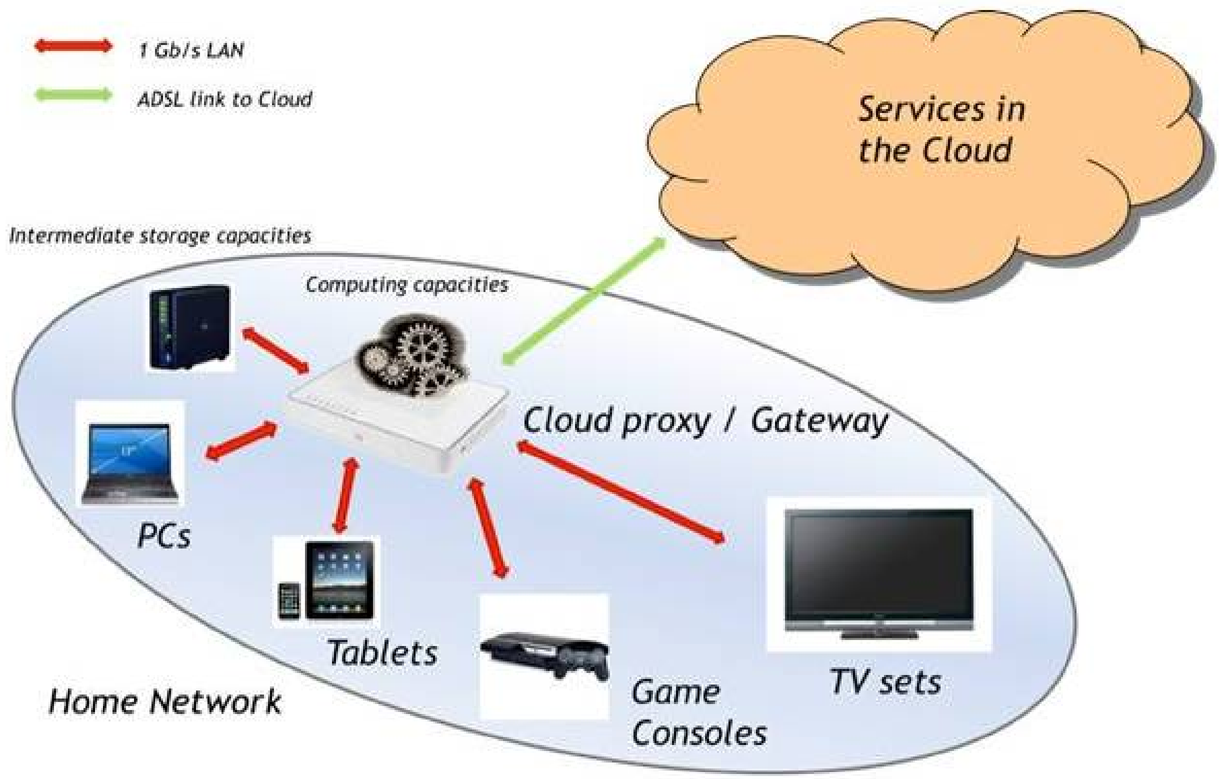
However, the link between the cloud and the end consumers (as well as many SMEs) appears to be a weak point of the system: it sometimes may be unique (therefore a single point of failure) and it may offer a relatively low bandwidth in some scenarios. Actually, typical ADSL bandwidths are in the range 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.



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.

Gaming feature support

Gaming functions are well suited to cloud computing because of computing power required and multi-user aspects. However, some acceleration features may be advantageously supported in a virtual machine running on the could proxy. 3D functions may also be supported in the cloud proxy although, mainly due to cost considerations, this may only possible in some years. Lastly, cloud proxy gaming functions may also take benefit of display facilities available in the home network, as far as no video output is directly present on the cloud proxy device.

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.

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.

Resource Monitoring

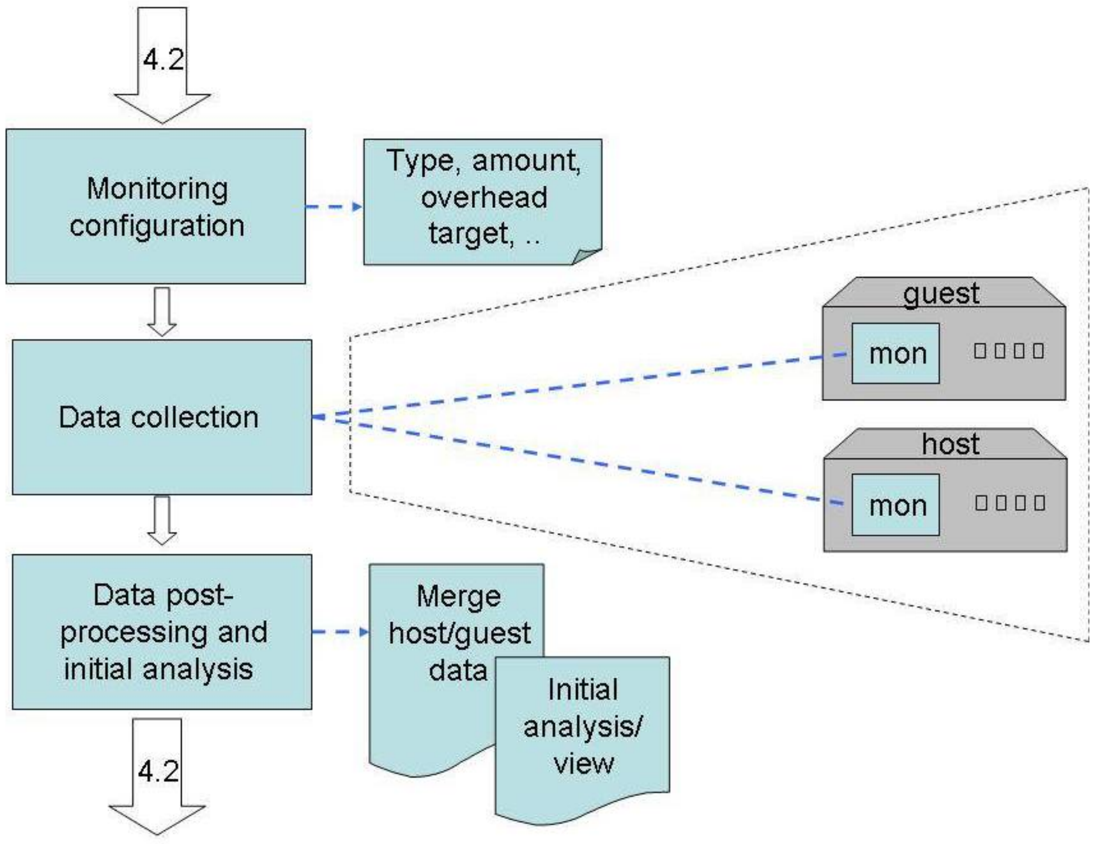
Target usage

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.

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.



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.

Question Marks

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

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?

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?

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.

References

[VMWare 09]	VMWare. vCloud API Programming Guide, Version 0.8.0. Online resource., 2009. [1] [1]
[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 [2], David Breitgand, A. Epstein [3], D. Hadas [4], I. Loy [5], Kenneth Nagin [6], J. Tordsson [7], C. Ragusa [8], M. Villari [9], Stuart Clayman [10], Eliezer Levy [11], A. Maraschini [12], Philippe Massonet [13], H. Muñoz [14], G. Tofetti [15]: Reservoir - When One Cloud Is Not Enough. IEEE Computer 44 [16](3): 44-51 (2011)
[Reservoir-Architecture2009]	Benny Rochwerger [2], David Breitgand, Eliezer Levy [11], Alex Galis [17], Kenneth Nagin [6], Ignacio Martín Llorente [18], Rubén S. Montero [19], Yaron Wolfsthal [20], Erik Elmroth [21], Juan A. Cáceres [22], Muli Ben-Yehuda [23], Wolfgang Emmerich [24], Fermín Galán [25]: The Reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development 53 [26](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 [23], David Breitgand, Michael Factor [27], Hillel Kolodner [28], Valentin Kravtsov [29], Dan Pelleg [30]: NAP: a building block for remediating performance bottlenecks via black box network analysis. ICAC 2009 [31]: 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

References

[1] http://communities.vmware.com/static/vcloudapi/vCloud
[2] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/r/Rochwerger:Benny.html
[3] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/e/Epstein:A=.html
[4] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/h/Hadas:D=.html
[5] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Loy:I=.html
[6] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/n/Nagin:Kenneth.html
[7] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/t/Tordsson:J=.html
[8] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/r/Ragusa:C=.html
[9] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/v/Villari:M=.html
[10] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/c/Clayman:Stuart.html
[11] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Levy:Eliezer.html
[12] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/m/Maraschini:A=.html
[13] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/m/Massonet:Philippe.html
[14] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/m/Mu=ntilde=oz:H=.html

[15] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/t/Tofetti:G=.html
[16] http://www.informatik.uni-trier.de/~ley/db/journals/computer/computer44.html
[17] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/g/Galis:Alex.html
[18] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Llorente:Ignacio_Mart=iacute=n.html
[19] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/m/Montero:Rub=eacute=n_S=.html
[20] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/w/Wolfsthal:Yaron.html
[21] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/e/Elmroth:Erik.html
[22] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/c/C=aacute=ceres:Juan_A=.html
[23] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/b/Ben=Yehuda:Muli.html
[24] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/e/Emmerich:Wolfgang.html
[25] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/g/Gal=aacute=n:Ferm=iacute=n.html
[26] http://www.informatik.uni-trier.de/~ley/db/journals/ibmrd/ibmrd53.html
[27] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/f/Factor:Michael.html
[28] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/k/Kolodner:Hillel.html
[29] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/k/Kravtsov:Valentin.html
[30] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/p/Pelleg:Dan.html
[31] http://www.informatik.uni-trier.de/~ley/db/conf/icac/icac2009.html

FI-WARE Data/Context Management

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 1.

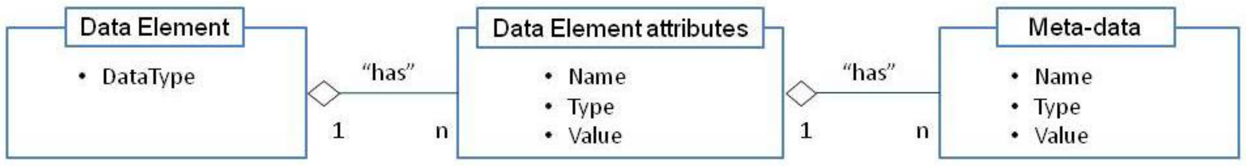


Figure 1: Data Element Structure

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 2.

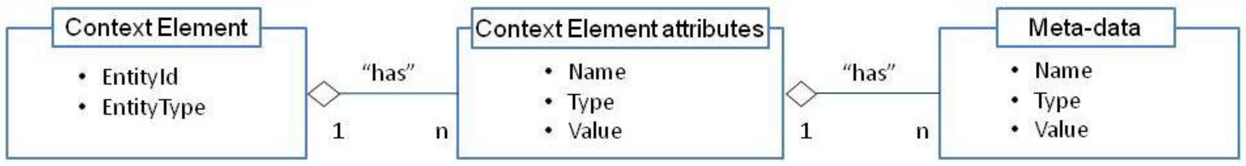


Figure 2: Context Element Structure Model

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 3 below 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.
- **Location 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.

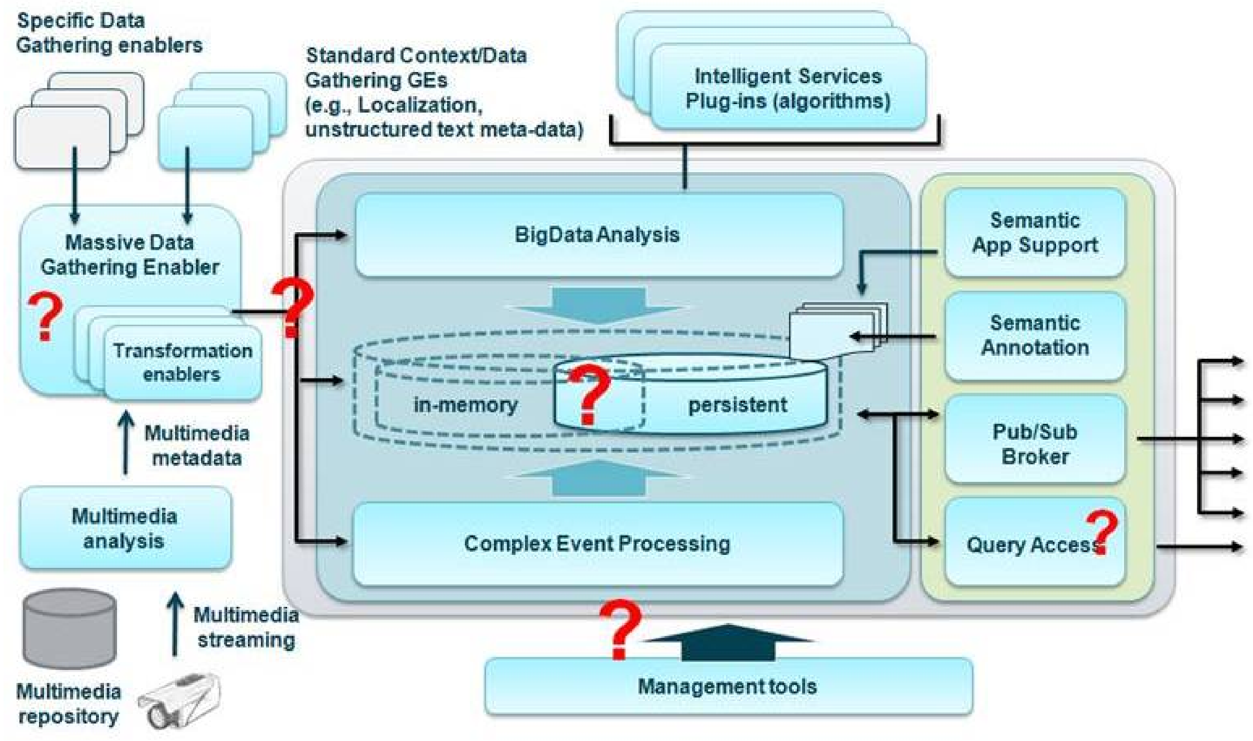


Figure 3: High-Level view of GEs in the Data/Context Management chapter

In addition to these GEs, the opportunity to include added value Generic Enablers implementing Intelligent Services in FI-WARE has been identified. However, their inclusion will depend very much on the identification of a demand from first users of FI-WARE and their potential to be common to a large number of applications in different domains. These enablers would be programmed based on the off-line and real time processing Generic Enablers, as well as on 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.

Generic Enablers

Publish/Subscribe Broker

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. 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.

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 4 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.

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 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.

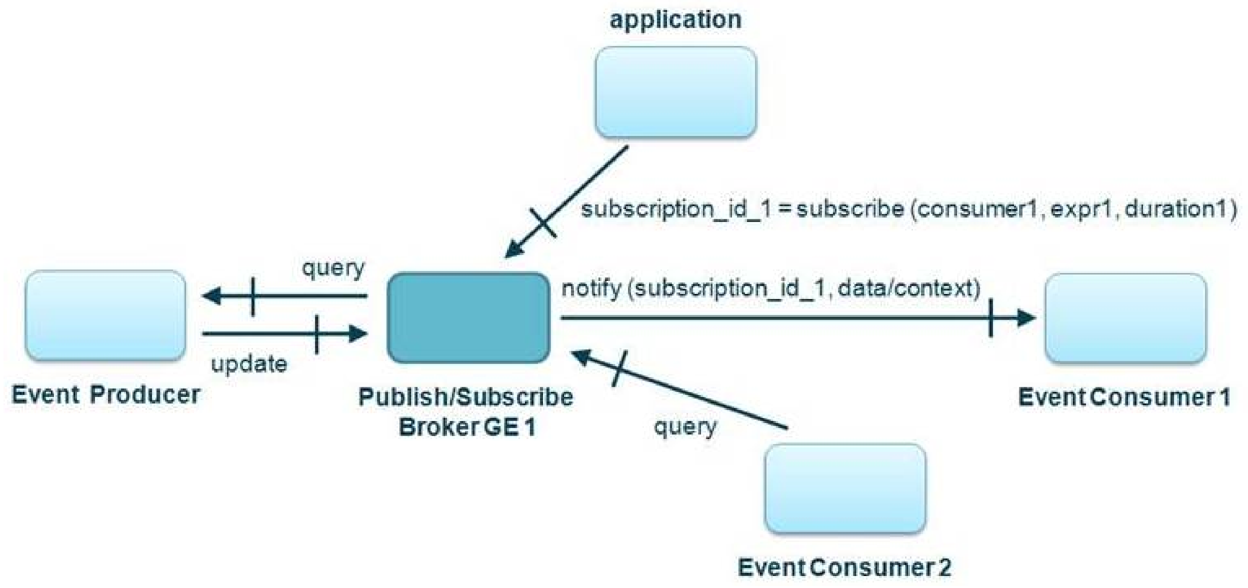

application
subscription_id_1 = subscribe (consumer1, expr1, duration1)
query
update
notify (subscription_id_1, data/context)
Event Producer
Publish/Subscribe Broker GE 1
query
Event Consumer 1
Event Consumer 2

Figure 4: 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 .

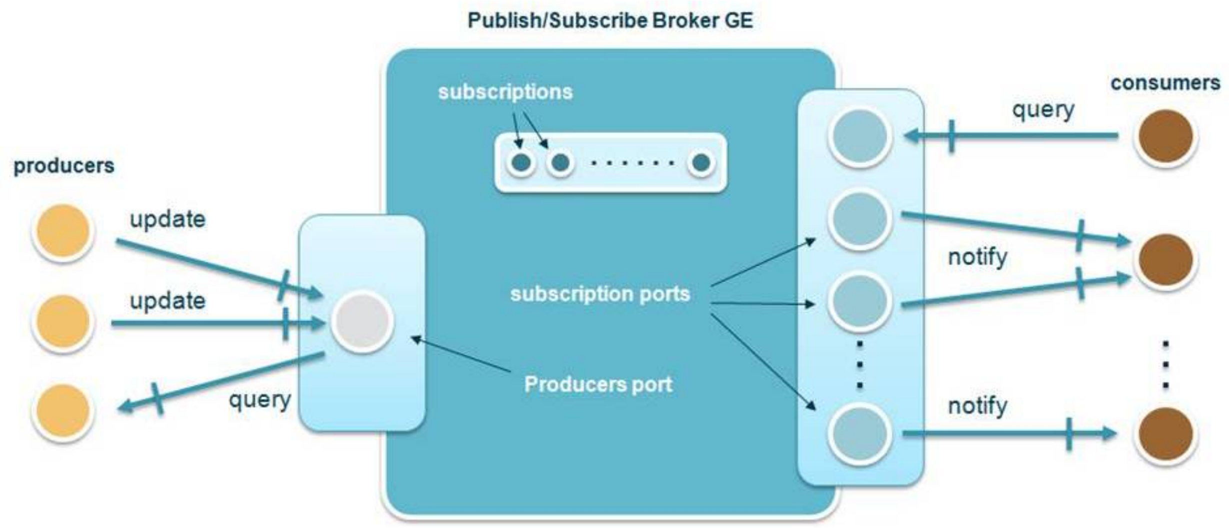


Figure 5: Multiple Event Producers and Consumers connected to a Publish/Subscribe Broker GE

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.

The preference given to OMA's framework, compared to other frameworks/technologies, is based on the following technical points:

- Ability to keep memory of events while conditions for the duration of these event hold, independently of who connects as Event Consumer. This will ease programming of Applications that may shutdown or are not initially ready but need to (re)start or synchronize processing of events that have been generated since a given point in time. Other frameworks such as WS-Notification will typically imply that the Broker receives events and pass them to Event Consumers that are connected at that time, without keeping events on memory (i.e., events received at the Broker are "consumed" by the Event Consumers connected at that moment).
- Suitability for a wide range of potential implementations of the Publish/Subscribe Broker GE, adaptable to run not only on traditional servers but small devices. This will ease the development of distributed networks of connected Publish/Subscribe Brokers running both on top of small devices and on servers. FI-WARE has to support working in a federated model with different types of federations: hierarchical (local, global, etc.), functional (social, physical, etc.) and/or mash (peer-to-peer info exchange). The OMA NGSI model allows this by design.
- Ability to define several alternative bindings, being particularly suitable to adapt to a REST binding, which seems like more appropriate in FI-WARE comparing to other frameworks where only one bindings, not necessarily REST-based, such as SOAP, is mandatory. The REST communication is much more suitable for heterogeneous communications of resource (and energy) constrain devices such as IoT nodes and mobile phones, as far as the protocol REST “per se” is very simple and with little (useless) overhead, and payload can be any (text SMS SMPP, XMPP, SMIME/SIP, XML, an ASCII structure or any proprietary binary encoded payload depending on application.
- Adaptability to handle data with no predefined structure (i.e., mandatory fields)
- Flexible subscription query language (being able to adapt for support of multiple query languages) compared to other frameworks such as WS-Notification which requires support of the notion of "topic".

On the other hand, ability to influence OMA direction towards changes in the specifications, when needed, is highly desirable. Other specifications/frameworks are considered more "stable" and due to legacy/backward-compatibility issues, less open to changes. FI-WARE partners have already contributed to OMA NGSI specs in the past and will be able to contribute in future.

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)

Complex Event Processing

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*.

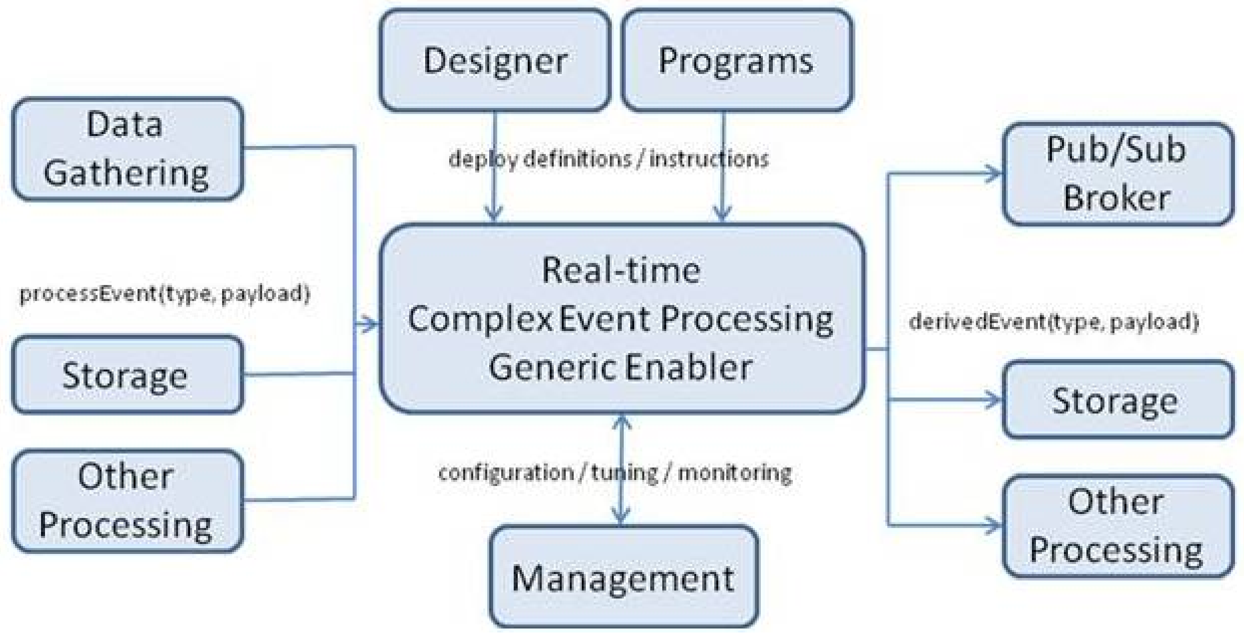

Designer
Programs
deploy definitions / instructions
Data Gathering
processEvent(type, payload)
Storage
Other Processing
Real-time Complex Event Processing Generic Enabler
Pub/Sub Broker
derivedEvent(type, payload)
Storage
Other Processing
configuration / tuning / monitoring
Management

Figure 6: Interactions with and APIs of the Real-time CEP Generic Enabler

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.

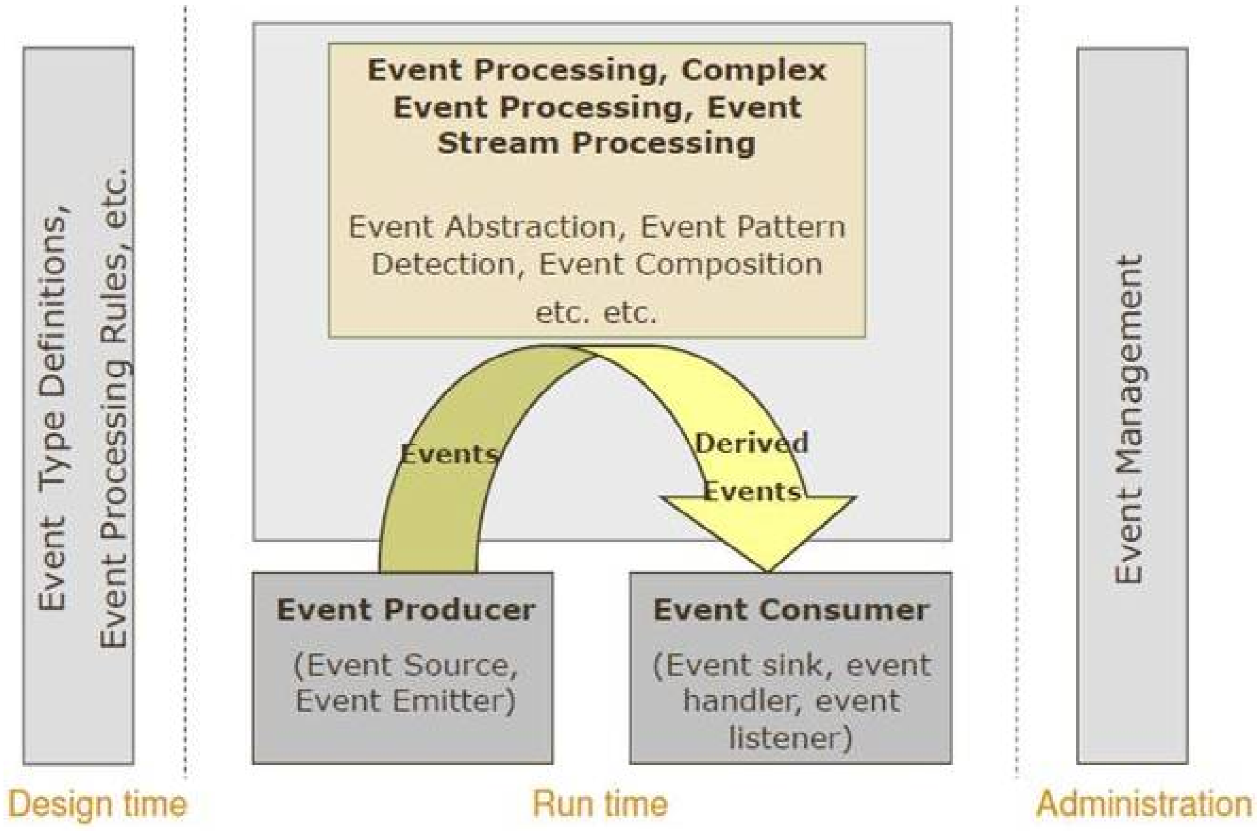

Event Type Definitions,
Event Processing Rules, etc.
Event Processing, Complex Event Processing, Event Stream Processing
Event Abstraction, Event Pattern Detection, Event Composition etc. etc.
Events
Derived Events
Event Producer
(Event Source, Event Emitter)
Event Consumer
(Event sink, event handler, event listener)
Event Management
Design time
Run time
Administration

Figure 7: 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.

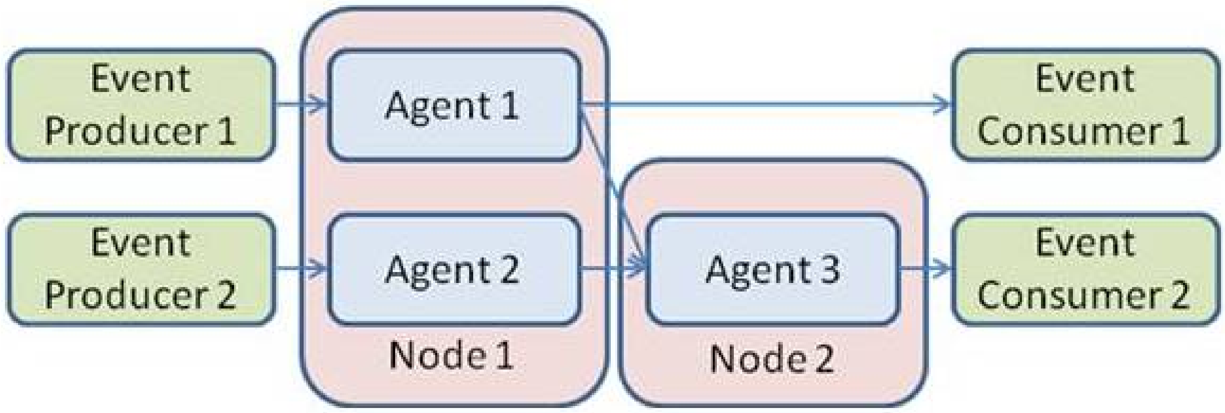


Figure 8: Illustration of an Event Processing Network made of producers, agents and consumers

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.

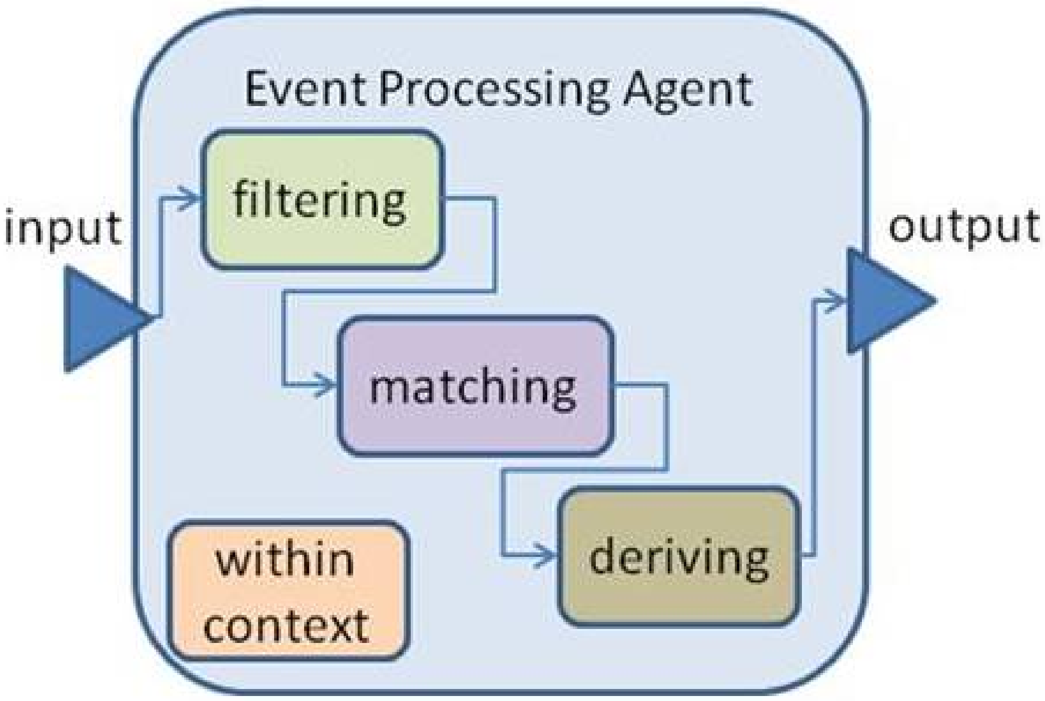


Figure 9: The building block of Event Processing Agent – simpler specification of the logic

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.

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

Big Data Analysis

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).

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.

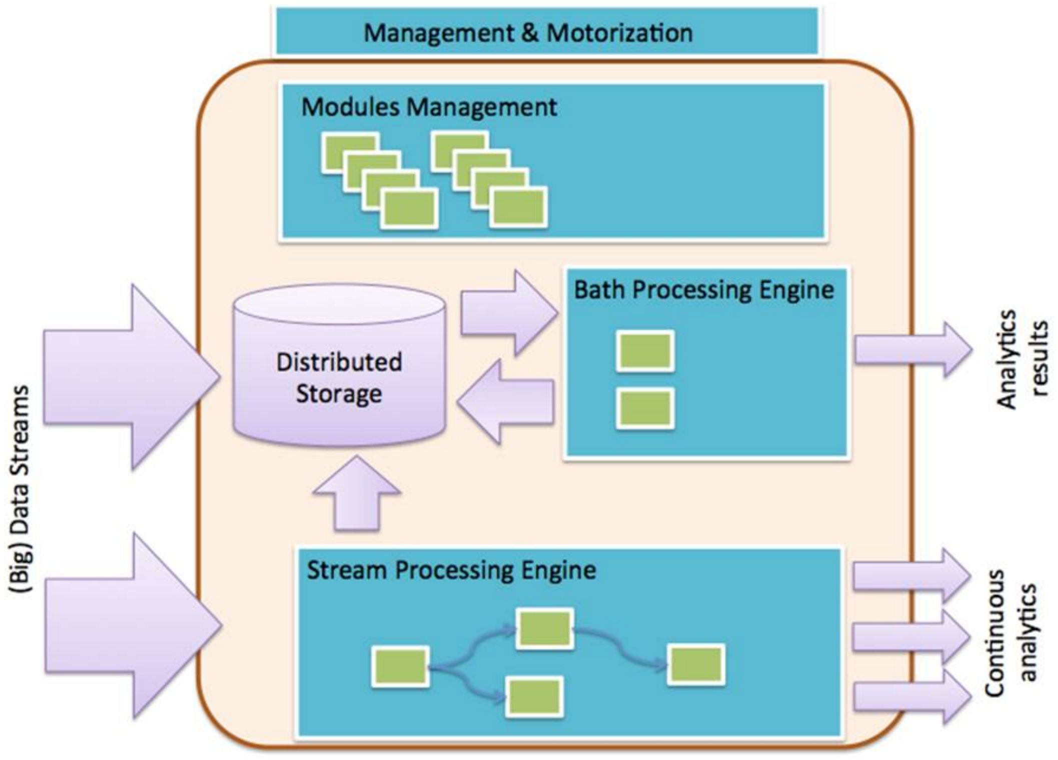


Figure 10: Big Data Analysis GE

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.

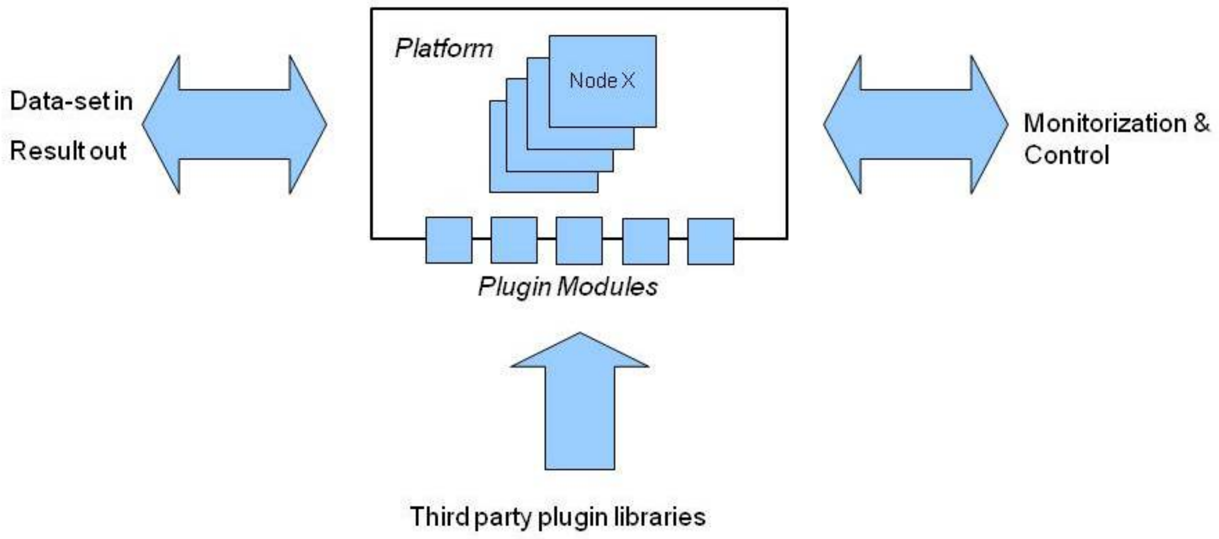


Figure 11: Big-data Platform

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.

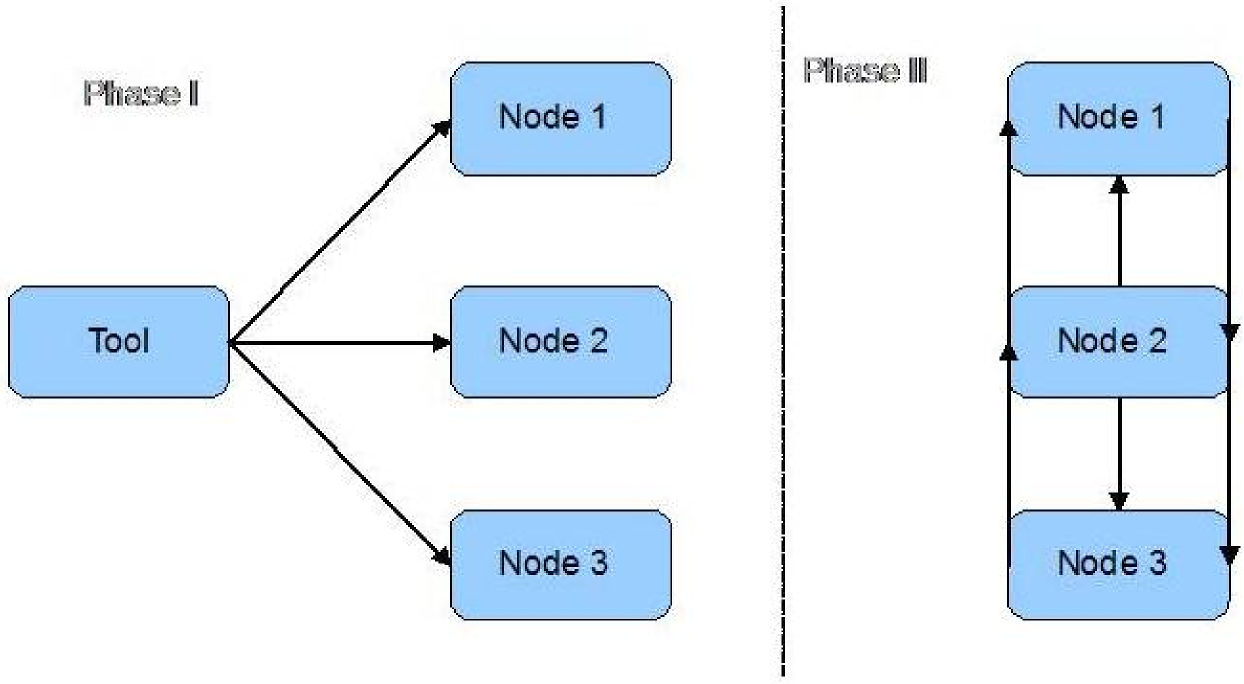


Figure 12: The two phases during upload of data to the platform

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.

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

Multimedia analysis

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, …).

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.

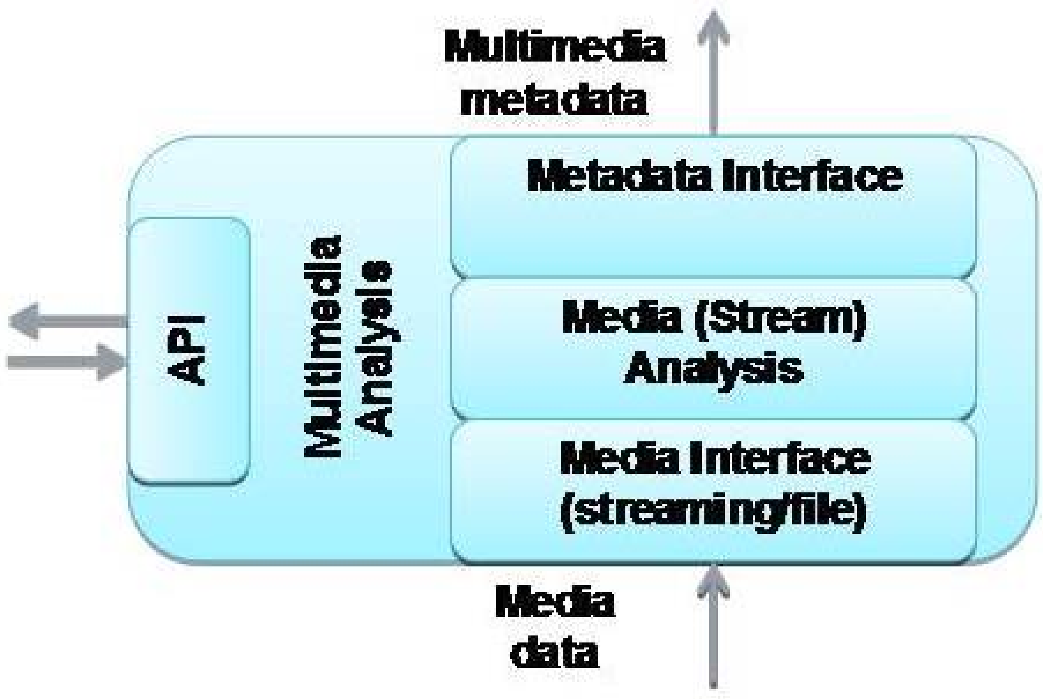


Figure 13: Multimedia Analysis GE – Generic description

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 14 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.

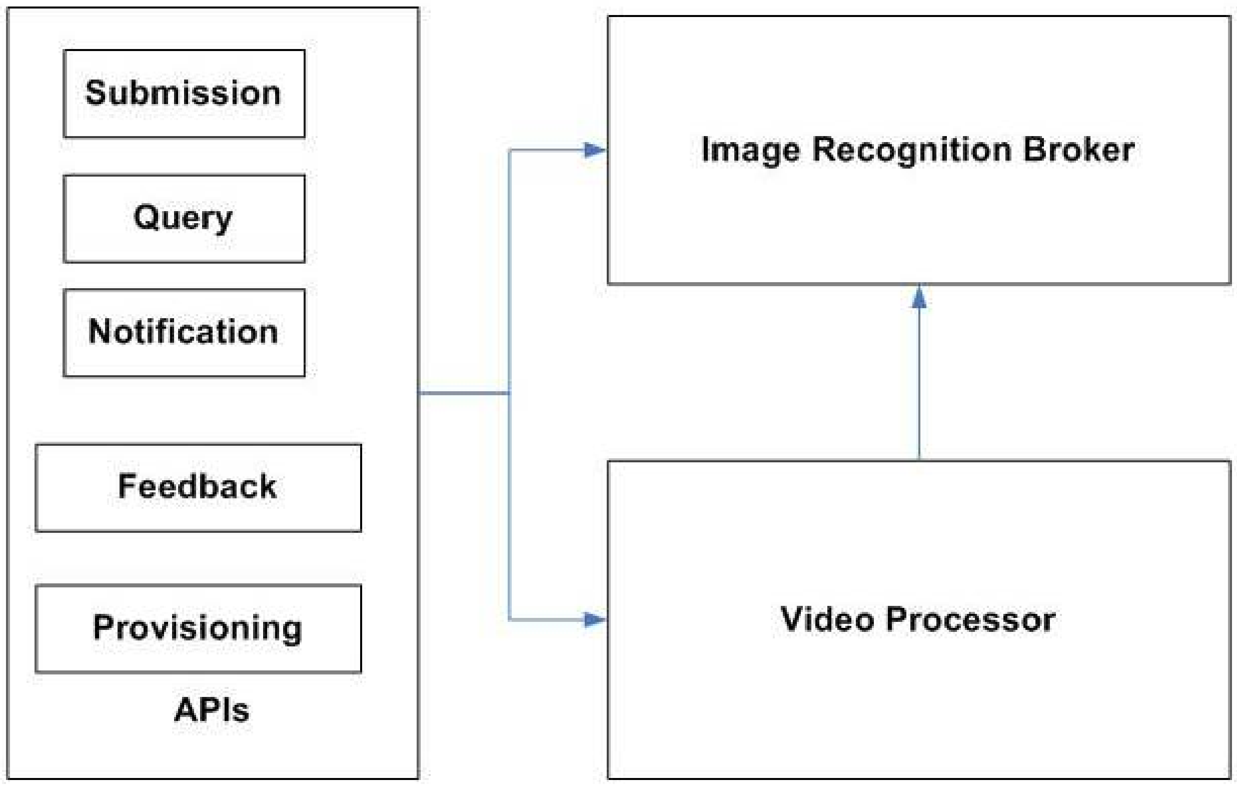


Figure 14: Multimedia Analysis GE – Generic description with detailed interfaces

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 15 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.

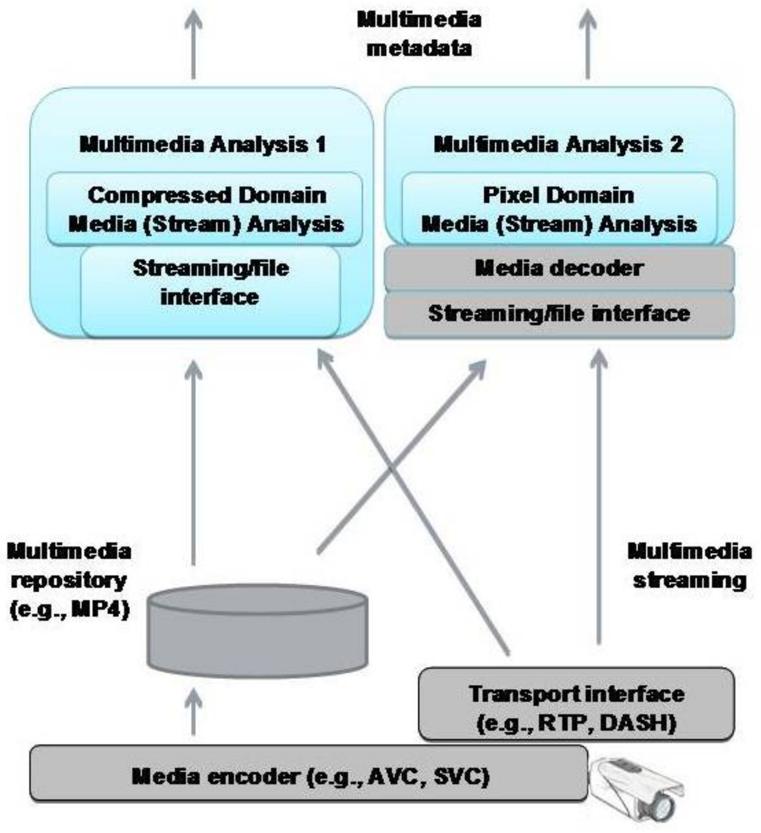

Multimedia metadata
Multimedia Analysis 1
Compressed Domain Media (Stream) Analysis
Streaming/file interface
Multimedia Analysis 2
Pixel Domain Media (Stream) Analysis
Media decoder
Streaming/file interface
Multimedia repository (e.g., MP4)
Multimedia streaming
Transport interface (e.g., RTP, DASH)
Media encoder (e.g., AVC, SVC)

Figure 15: Multimedia Analysis GE – Example realizations

Figure 16 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).

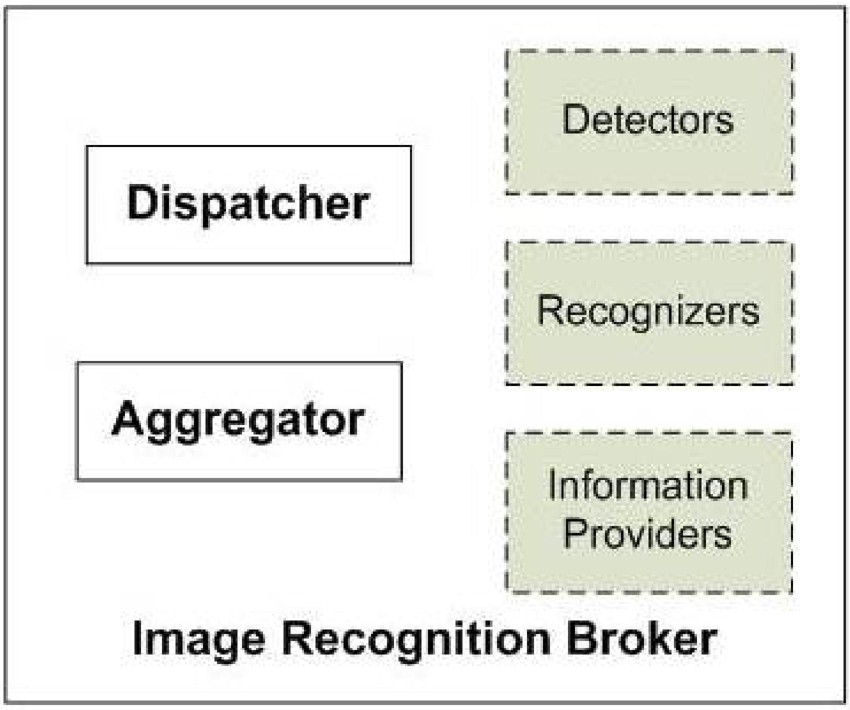


Figure 16: Multimedia Analyser - Image Recognition Broker

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.

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

Unstructured data analysis

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.

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.

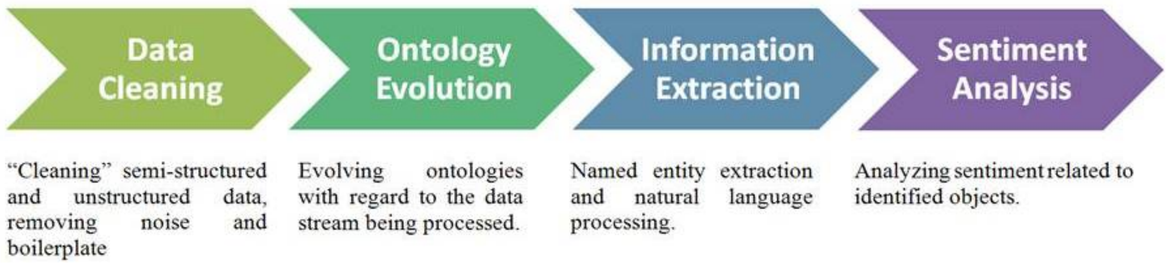

Data Cleaning
Ontology Evolution
Information Extraction
Sentiment Analysis
"Cleaning" semi-structured and unstructured data, removing noise and boilerplate
Evolving ontologies with regard to the data stream being processed.
Named entity extraction and natural language processing.
Analyzing sentiment related to identified objects.

Figure 17: 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.

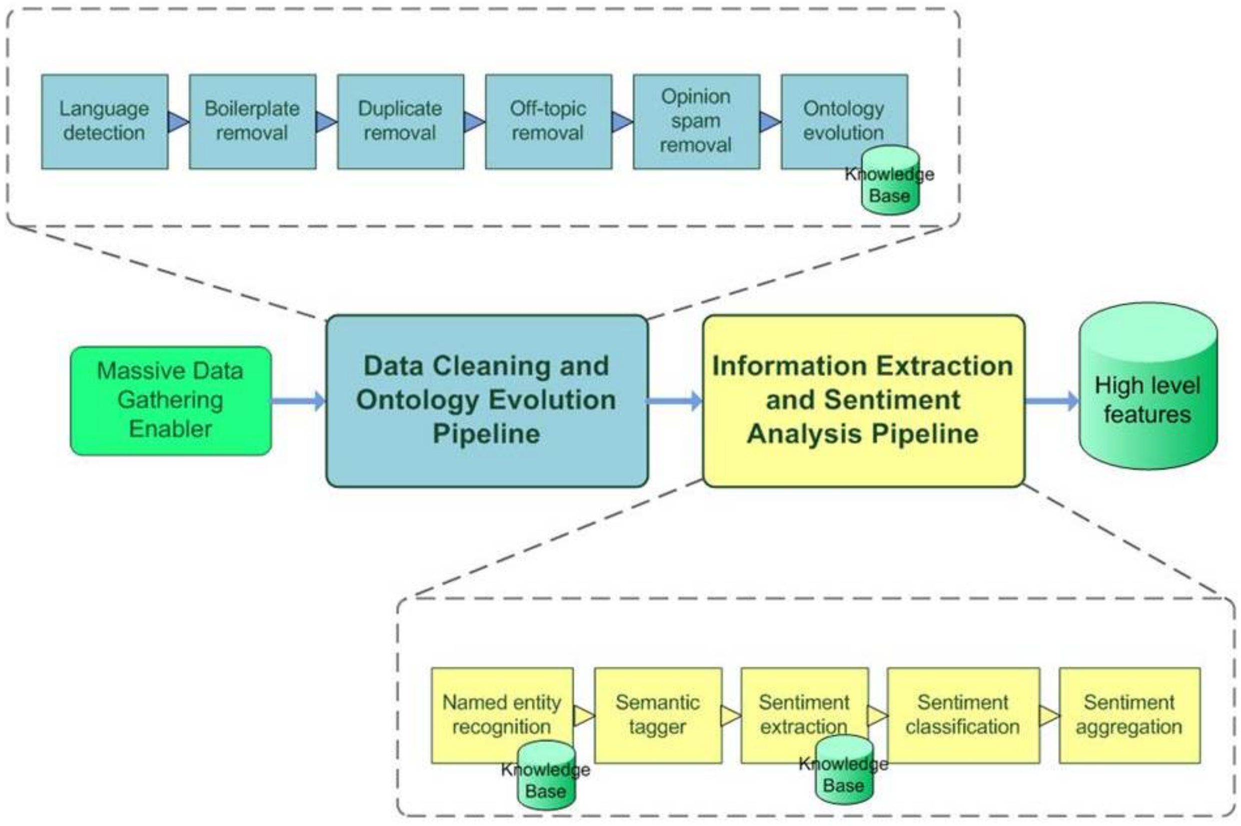


Figure 18: Architecture of Unstructured data processing enable pipeline

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.

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.

Meta-data Pre-processing

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.

GE description

Figure 19 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.

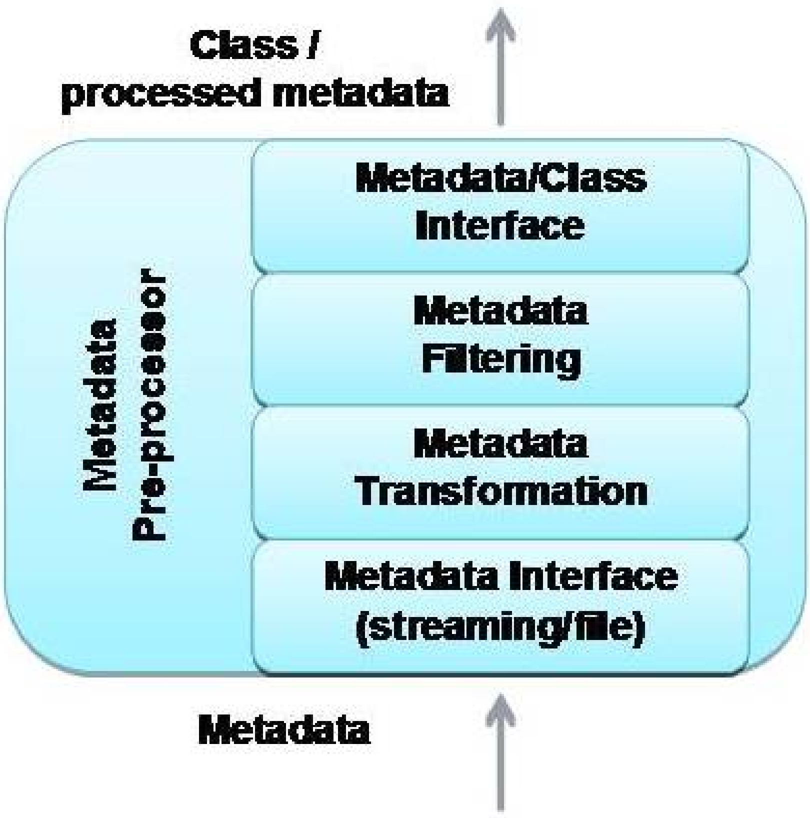


Figure 19: GE “Metadata Pre-Processor”

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 20 shows an example realization of the “Metadata Pre-Processor” Generic Enabler.

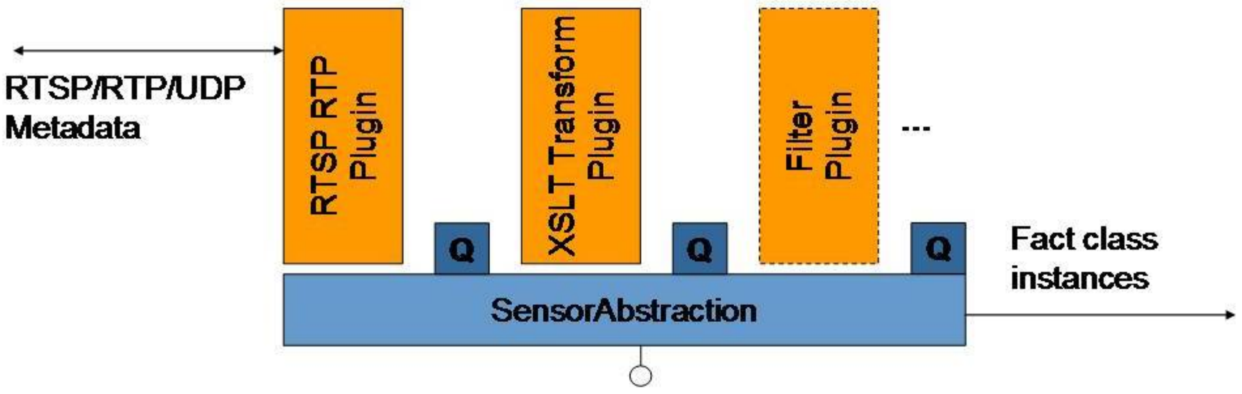


Figure 20: Example metadata pre-processing chain

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.

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).

Location Platform

Target usage

The Location GE in FI-WARE targets any application, GEs in FI-WARE, or any complementary platform enabler, that aims to retrieve mobile device positions and Location area events. The Location 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 Location 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.

GE description

The following figure describes the main modules of the Location 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).

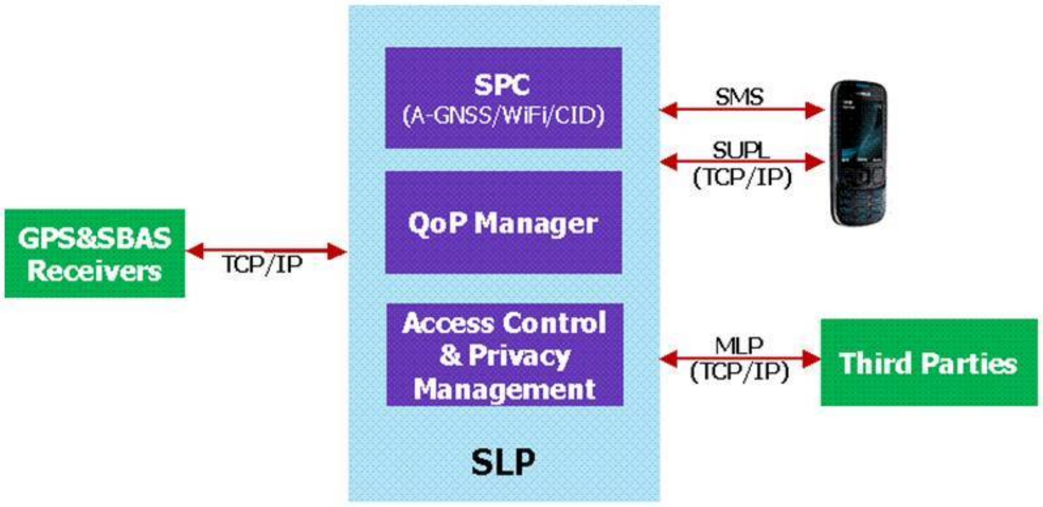


Figure 21: SUPL Location Platform

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:

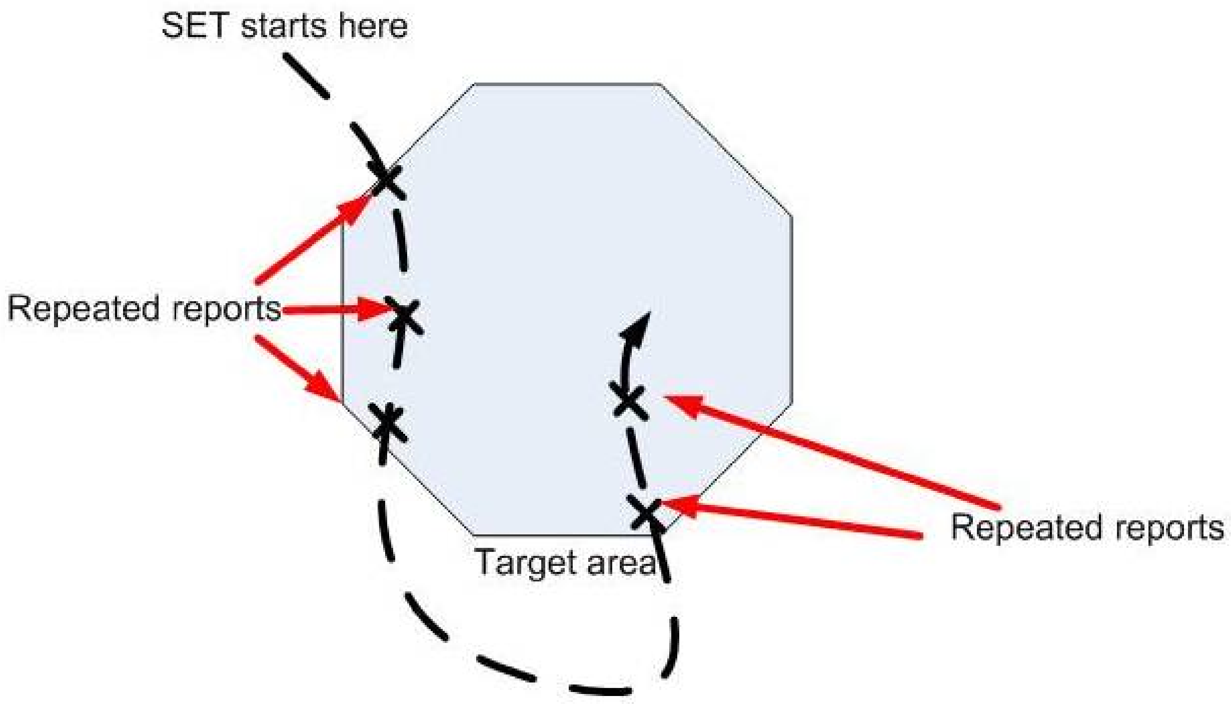


Figure 22: Event Triggers

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.

Query Broker

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.

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.

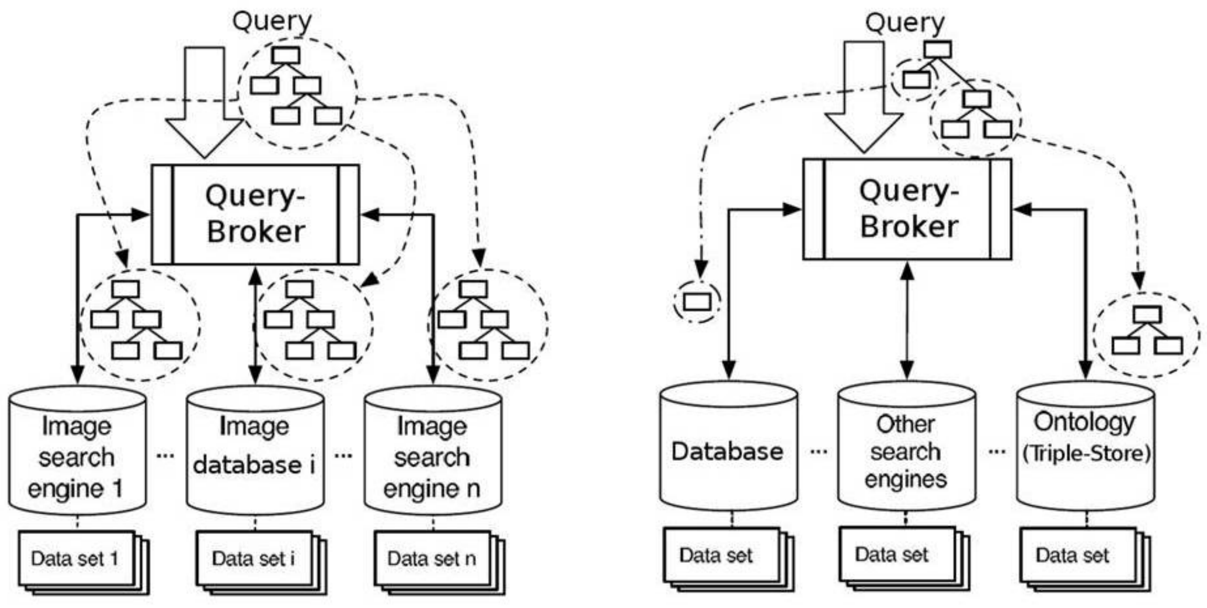
- Metadata format interoperability:

  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 23



(a) Local processing (b) Distributed processing

Figure 23: 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 24 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.

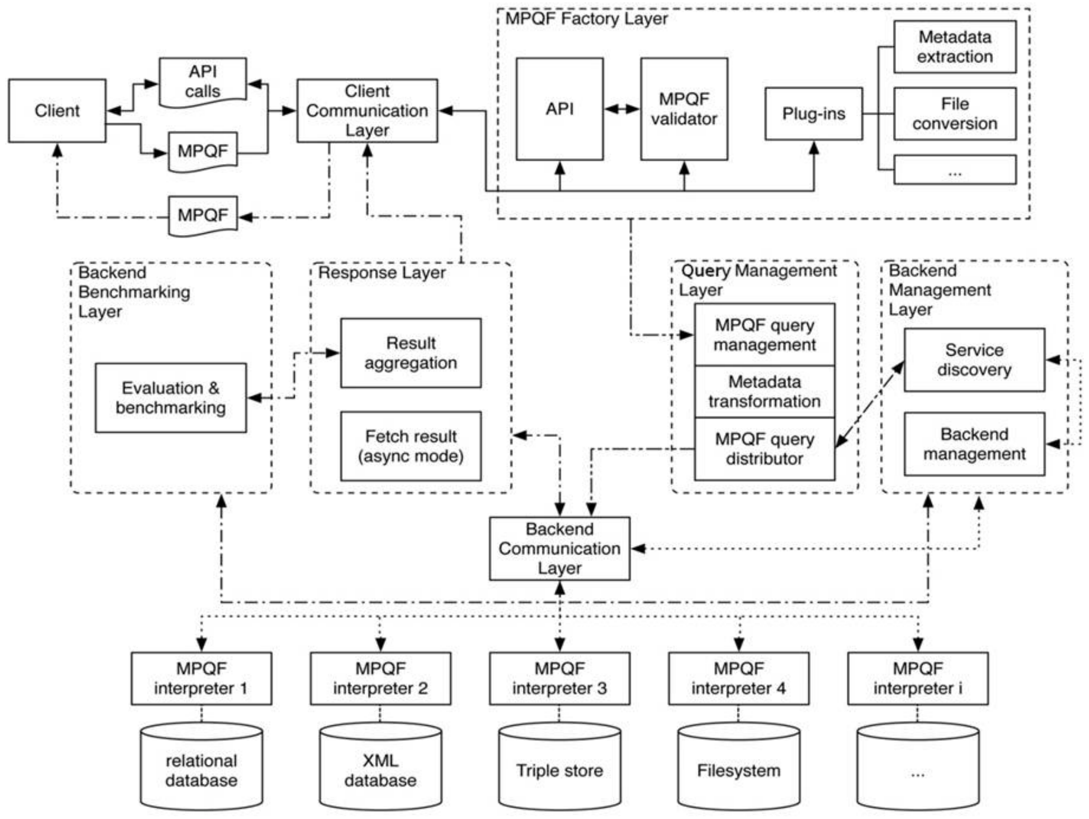


Figure 24: Architecture of the QueryBroker

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.

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

Semantic Annotation

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)

GE Description

Figure 25 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.

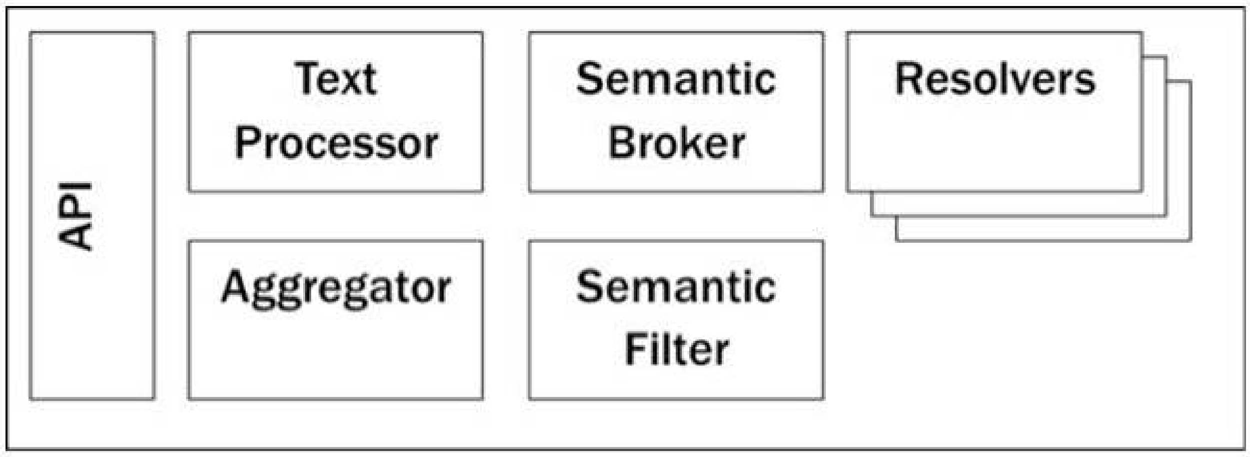


Figure 25: Semantic Annotation GE

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 26 shows an example realization of the “Semantic Annotator” Generic Enabler.

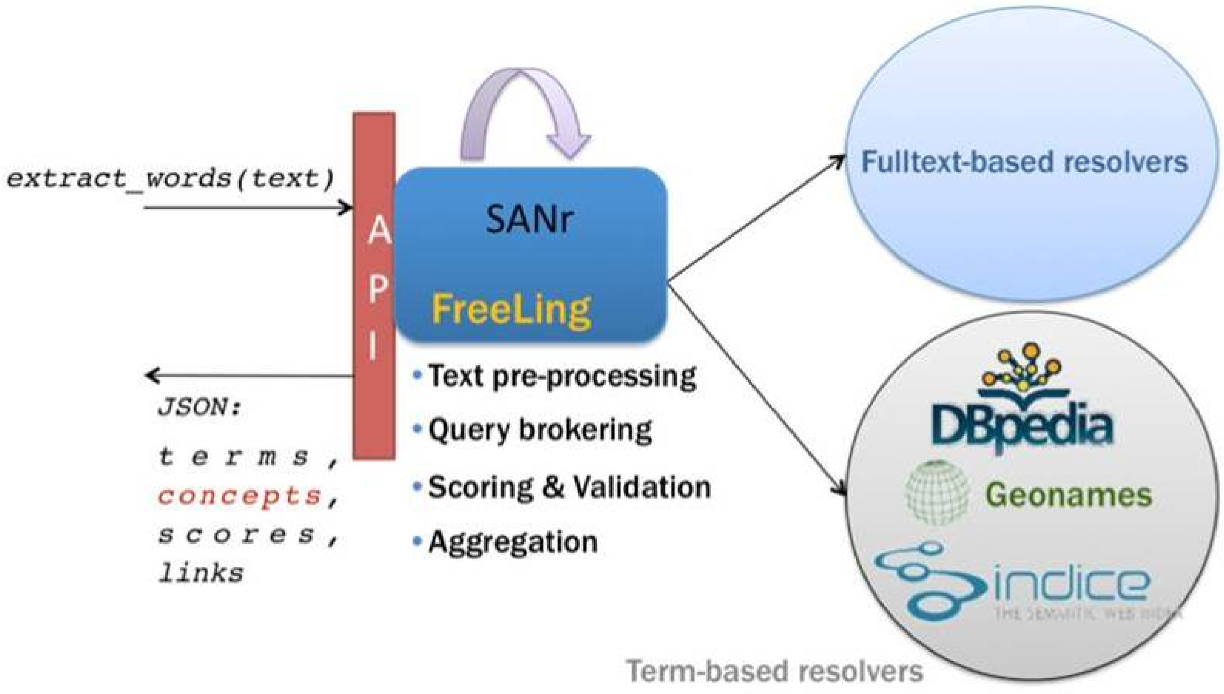


Figure 26: Example Semantic Annotator

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.

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

Semantic Application Support

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.

GE Description

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 27 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.

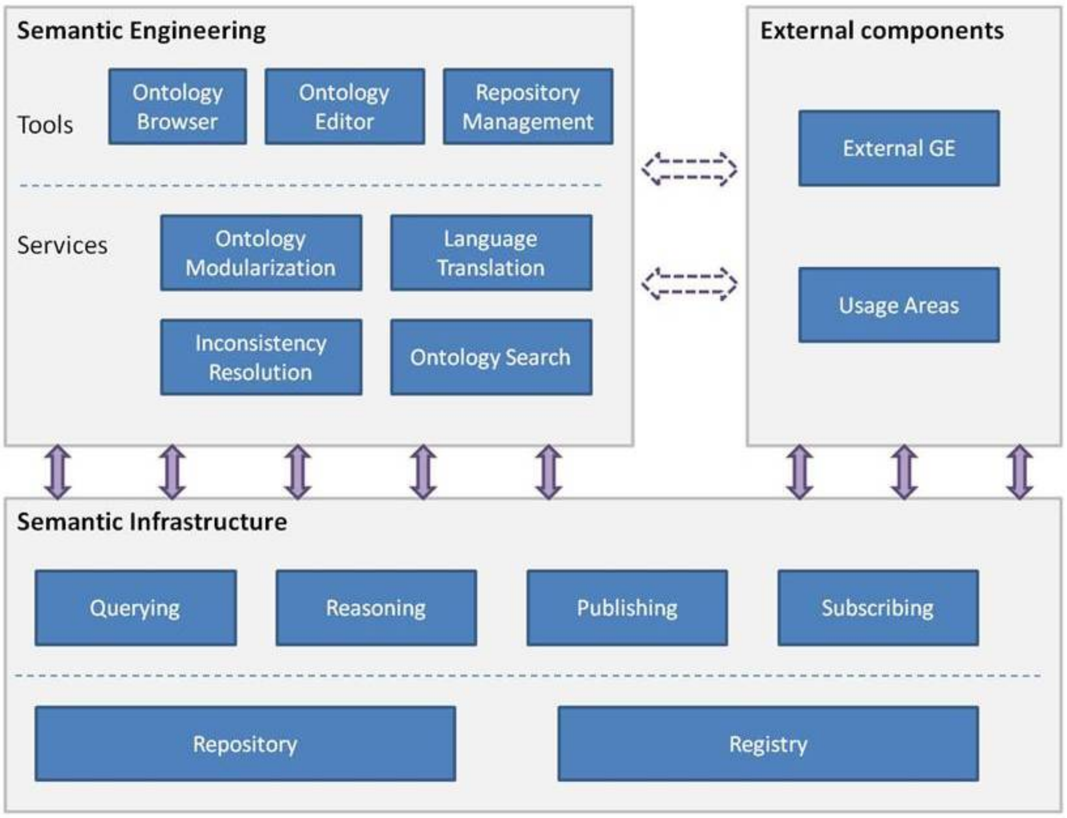


Figure 27: Semantic Web Application Support Enabler architecture

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.

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.

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).

Social Network Analysis

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.

Critical product attributes

- The Social Network Analytical capabilities provided by the service integrate social communications of different nature (on-line social networks and mobile voice 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.

Mobility Analysis

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.

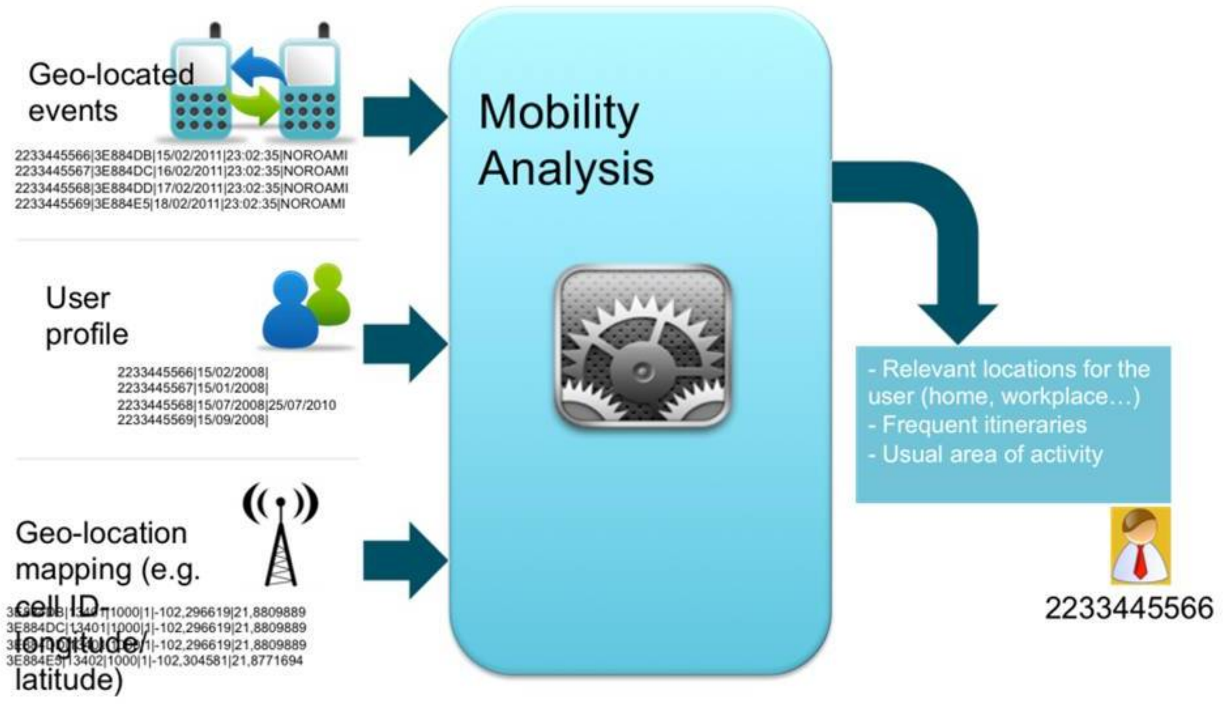

Geo-located events
2233445566|3E884DB|15/02/2011|23:02:35|NOROAMI
2233445567|3E884DC|16/02/2011|23:02:35|NOROAMI
2233445568|3E884DD|17/02/2011|23:02:35|NOROAMI
2233445569|3E884E5|18/02/2011|23:02:35|NOROAMI
User profile
2233445566|15/02/2008|
2233445567|15/01/2008|
2233445568|15/07/2008|25/07/2010
2233445569|15/09/2008|
Geo-location mapping (e.g. cell ID- longitude/ latitude)
3E884DC|13401|1000|1|-102,296619|21,8809889
3E884E5|13402|1000|1|-102,304581|21,8771694
Mobility Analysis
- Relevant locations for the user (home, workplace…)
- Frequent itineraries
- Usual area of activity
2233445566

Figure 28: 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.

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.

Real-time recommendations

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.

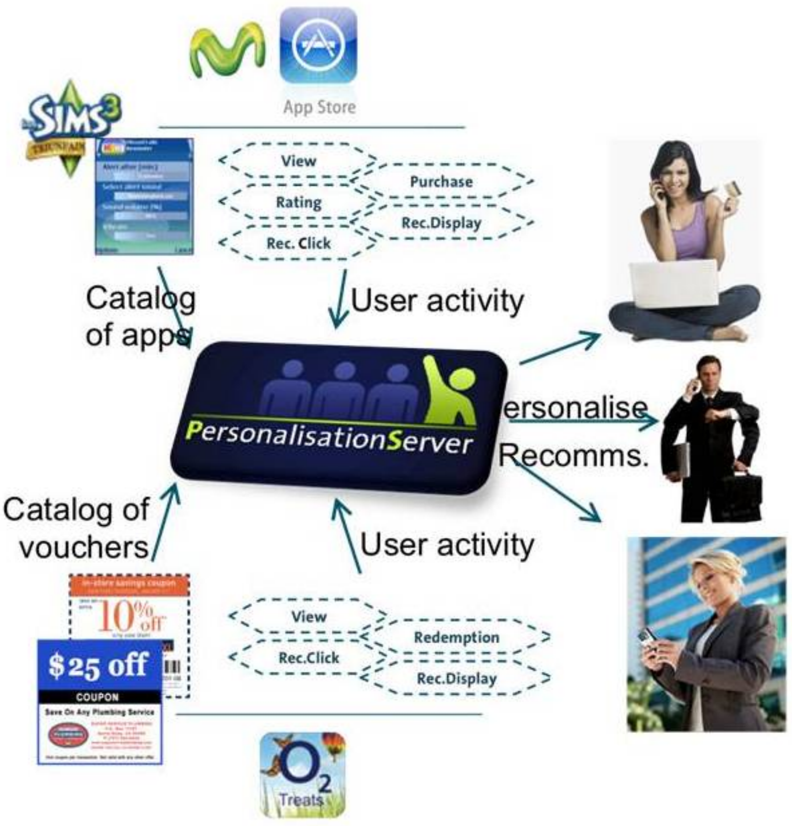


Figure 29: Recommendation system

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.

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

Web behaviour analysis for profiling

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.

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.

Opinion mining

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.

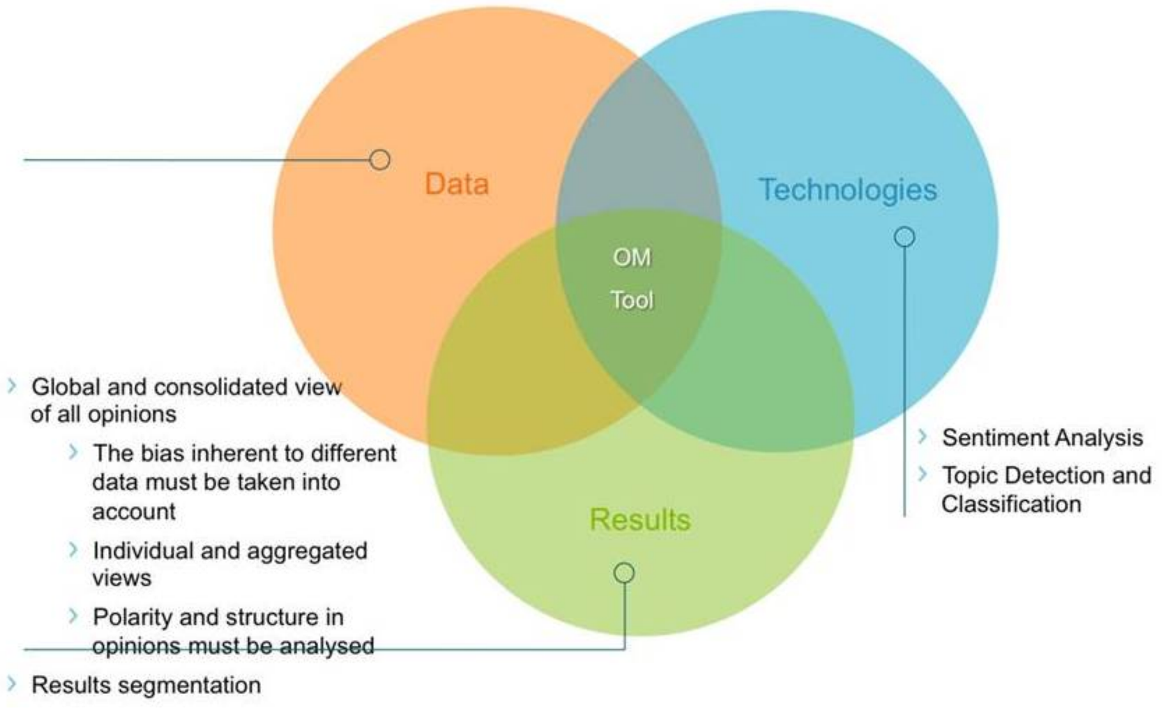

Data
Technologies
OM Tool
Results
› Global and consolidated view of all opinions
› The bias inherent to different data must be taken into account
› Individual and aggregated views
› Polarity and structure in opinions must be analysed
› Results segmentation
› Sentiment Analysis
› Topic Detection and Classification

Figure 30: Functionalities of the opinion mining intelligent services

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.

Question Marks

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

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.

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.

Location services

Some topics has been identified for further discussion regarding capabilities supported by the Location 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 Location 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 Location 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 Location 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.

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.

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.
[Chandy 11]	Mani K. Chandy, Opher Etzion, Rainer von Ammon, “10201 Executive Summary and Manifesto – Event Processing”, Schloss Dagstuhl, 2011.
[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.

[EPIA]	Opher Etzion and Peter Niblett, “Event Processing in Action”, Manning Publications, August, 2010, ISBN: 9781935182214
[EPTS]	Event Processing Technical Society: http://www.ep-ts.com/ [1]
[EPTS-RA 10]	Adrian Paschke and Paul Vincent. “Event Processing Architectures”, Tutorial, DEBS 2010. [2] [2]
[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. [3] [3]
[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

References

[1] http://www.ep-ts.com/
[2] http://www.slideshare.net/isvana/debs2010-tutorial-on-epts-reference-architecture-v11c
[3] 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

FI-WARE Internet of Things (IoT) Services Enablement

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*

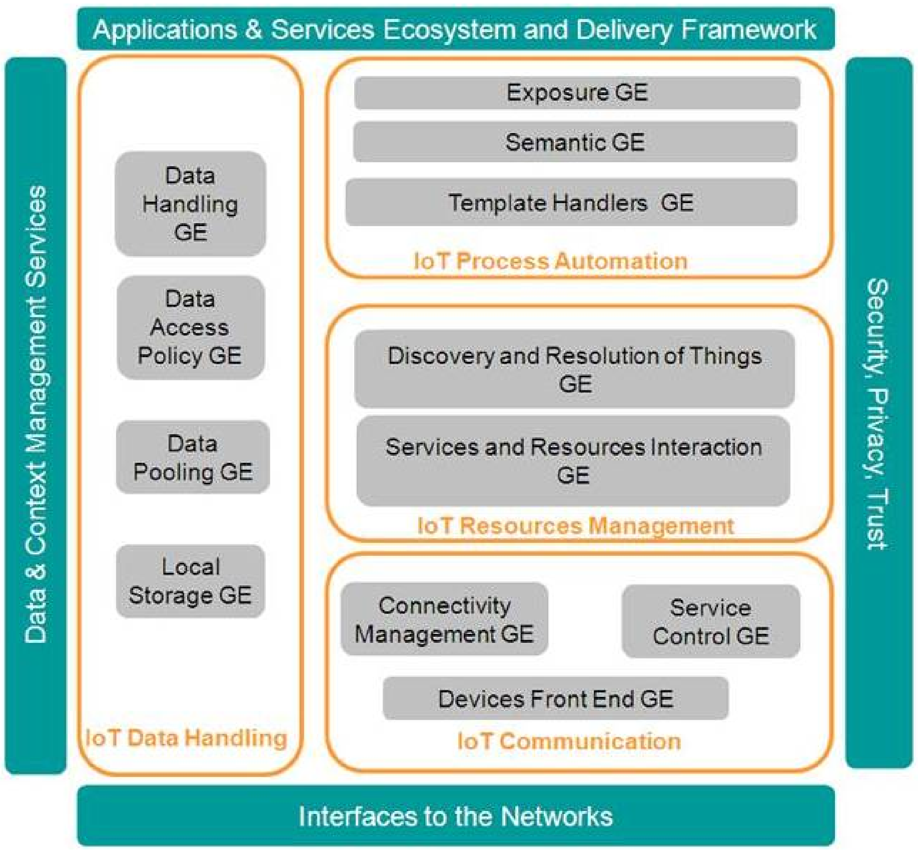
Tomorrow: David wants to go sooner to his office because the home clock screen advertises that weather forecast was too optimistic today and that bad weather will come during the night. The picture, based on 357 climate sensors, shows clearly that the snow storm will arrive around 7:30 AM. When he begins 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

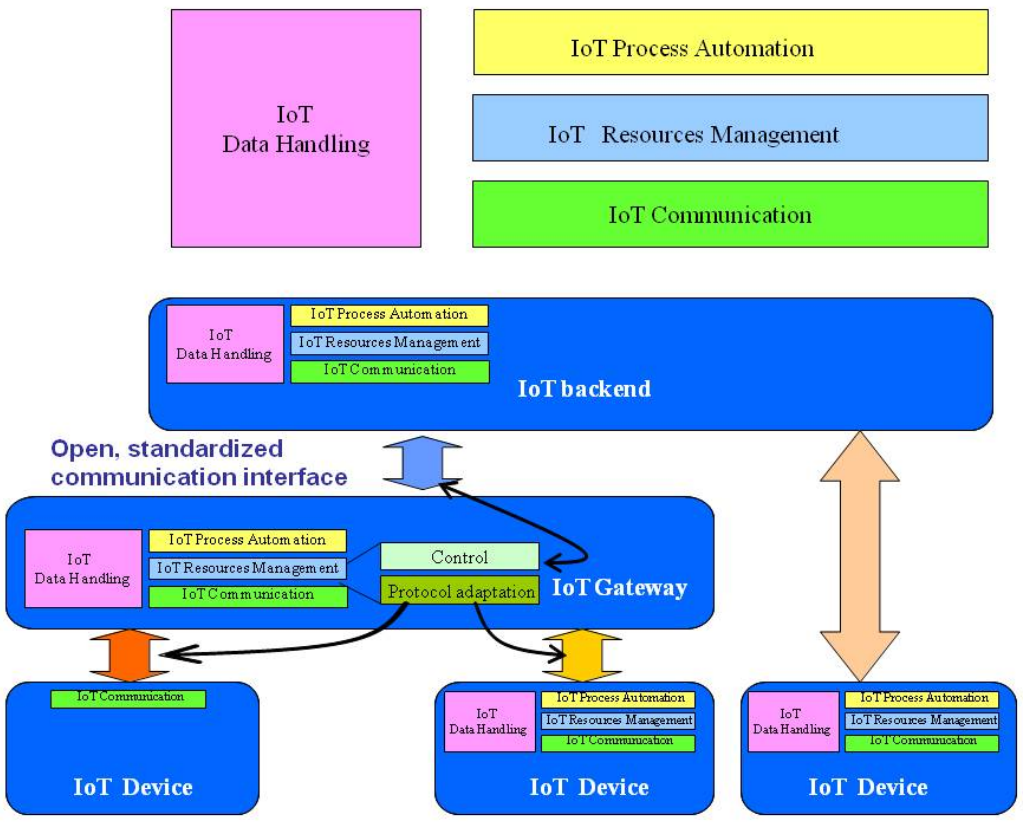


GEs of the IoT Service Enablement platform and its connections to other technical chapters

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 the previous figure, 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 the figure below.



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

There are a number of relations between GE-s in the Interface to Networks and Devices (I2ND) and IoT Services Enablement. The Cloud Edge (CE) GE from I2ND is able to locally execute downloadable applications and could be a relevant container for IoT gateways. In particular, CE may host gateway functionality for connecting other devices as mobile. In concrete terms cloud edge will provide mostly virtual machine images in which containers can be executed. CE will provide direct interface to these virtual machines/containers and will support the interface and protocols to a number of devices such as RFID, BT, assuming drivers exist for the associated operating system and relevant protocols/interfaces. The Connected Device Interface (CDI) GE functionality from I2ND might be also acting as hosting capability for the IoT gateway and/or device functionality: it is to be understood if scenarios in which this might occur are required for FI-WARE and/or UA projects. The basic idea is twofold regarding user endpoints (phones, tablets, etc) functionalities:

- have local sensor(s), in this sense they might act as gateways or proxies for these local sensors, exposing their functionality as an IoT gateway/proxy
- have local radio interfaces (Bluetooth, ZigBee, WiFi etc) which allows them to access devices/IoT resources in their neighborhood and expose their functionality as an IoT gateway.

The S3C GE from the I2ND technical chapter is relevant in case for 3GPP-based IoT devices.

- On one hand the S3C GE could be used for key exchange and validation of messages between IoT devices through 3GPP infrastructures
- Additional point of interest is the caching module, which could be interfaced from our perspective with the IoT Data Handling generic enablers. Device or gateway databases can be handled together with the caching module of the S3C enabler. Yet additionally the network event management component might catch and distribute different events (aggregated events, events from several IoT devices etc) than what one single IoT device can see, this could be used as an additional input for or synchronized with our micro database or cloud hosting databases, especially for real time data.
- The network connectivity management module could interact with IoT communication GE-s from the perspective of supporting intermittent connectivity and session management.

The following section elaborates on each of our 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.

IoT Communications

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.

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. Beyond communication layer protocols in the WSN domain the ISO/IEC CD 29182-1 standard [SNRA] aims to specify a sensor network reference architecture which needs to be taken into account when defining the architecture of the IoT technical chapter and its interaction with the other technical chapters in FI-WARE. Sensor Network Reference Architecture [SNRA] is to define an overall framework that will facilitate description and understanding of the interrelation and interaction between the components of sensor network applications and services.

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. In the M2M area beyond ETSI M2M a number of industry associations were formed which result in de facto interface protocols being specified, created and supported by a number of devices. One such example is the ONVIF forum [ONVIF] that specifies a global standard for the interface of IP-based physical security products. ONVIF is committed to the adoption of IP in the security market. The ONVIF specification will ensure interoperability between products regardless of manufacturer. Another such example is the OpenMeter consortium [OpenMeter] which specifies a set of open and public standards for AMI, supporting electricity, gas, water and heat metering. Another interesting consortium, the Open Geospatial Consortium, has described some relevant technical specifications, mostly well known as Sensor Web Enablement (SWE) services [OGC SWE] which are designed to enable discovery of sensor assets and capabilities, access to those resources and data retrieval, as well as subscription to alerts, and tasking of sensors to control observations. OGC SWE does not have architecture in a strict sense. It defines more collaboration scenarios for the interfaces or languages and protocols. It is important to emphasise that Sensor Markup Language (SensorML) is the central element of the OGC SWE and include definition of the information model or the protocol description and how the various services are accessed.

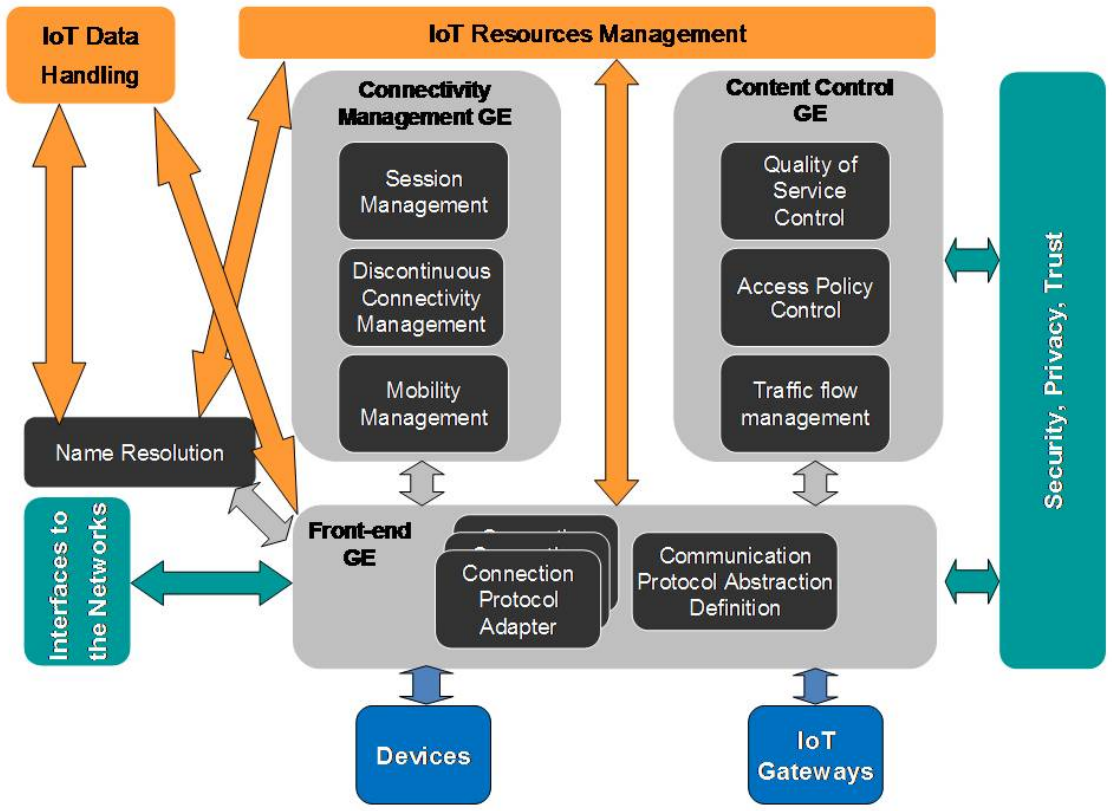
IoT communication Generic Enablers will utilize the above-mentioned technologies that support communications between distributed things and devices (e.g. gateway and middleware solutions) and will develop standard interfaces and interoperable solutions to coordinate the communication and interaction 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. -

The figure below 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)



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.

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

IoT Resources Management

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.

Description of GEs

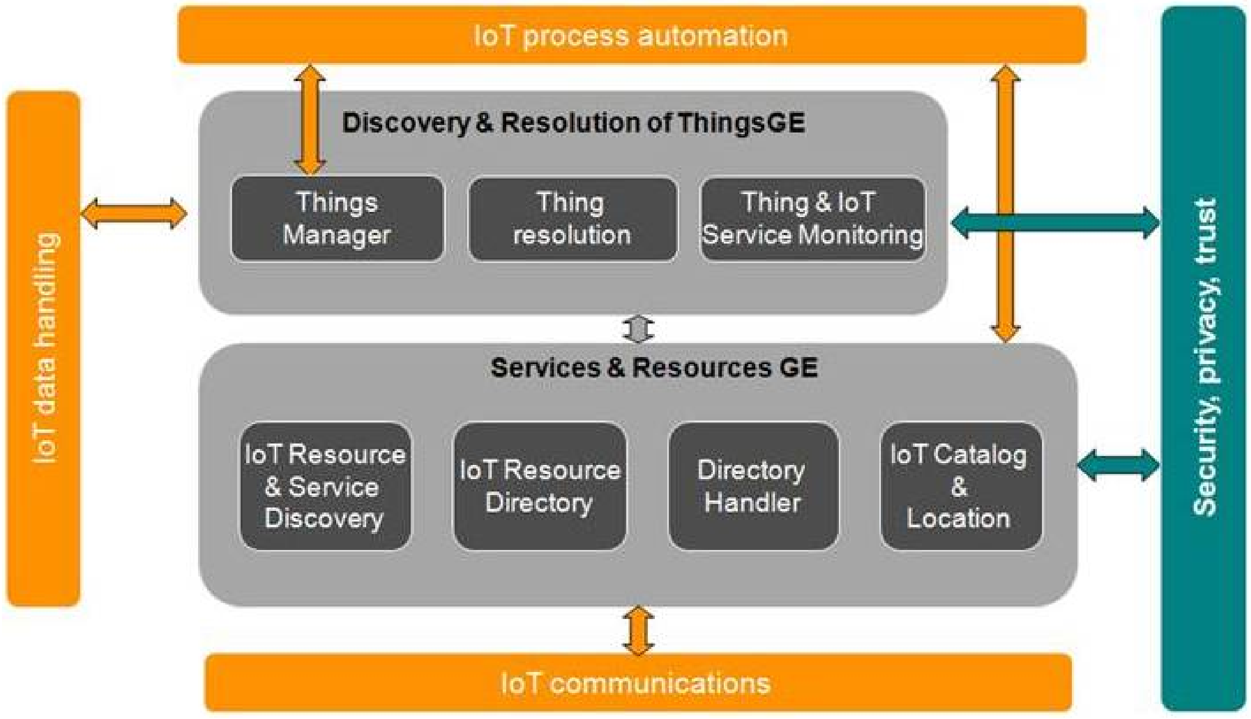
The figure below shows the main Generic Enablers implementing IoT Resource Management functions. These Generic Enablers will provide altogether at least the following functions:

- 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.



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.

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

IoT Data handling

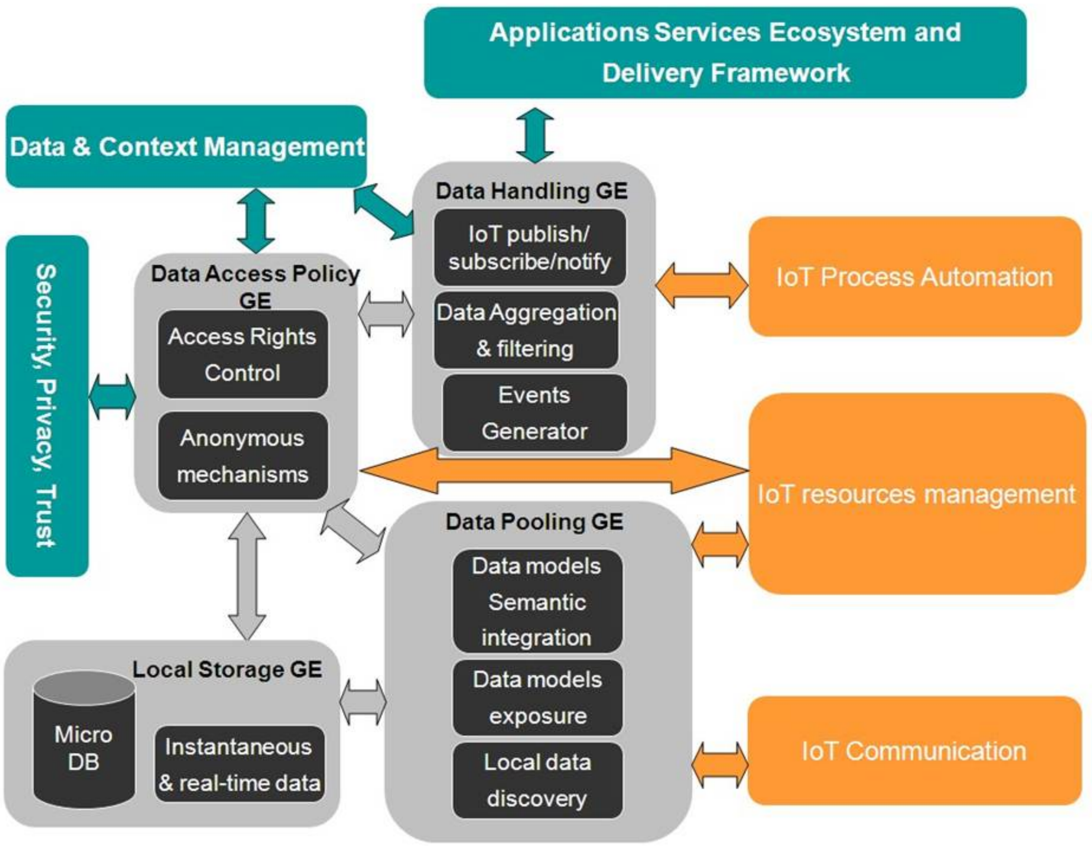
Target usage

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.



Architecture of IoT Data Handling and its 4 GEs

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.

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

IoT Process Automation

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.

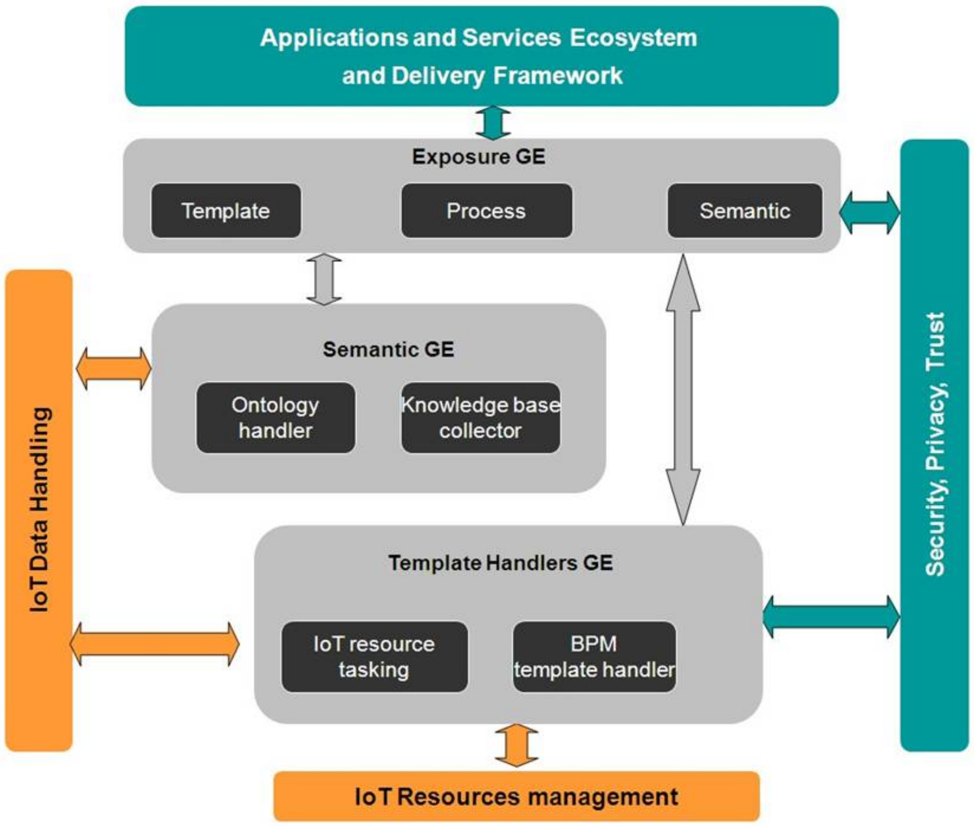
A necessary step for improving IoT processes in large scale solutions is the ability to combine data generated by IoT with business processes that are responsible for scheduling and control. For IoT scenarios, it is necessary to devise a reasonable and complete modelling construct meaning that the process 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

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.



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

Question Marks

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.

IoT Data as a Service

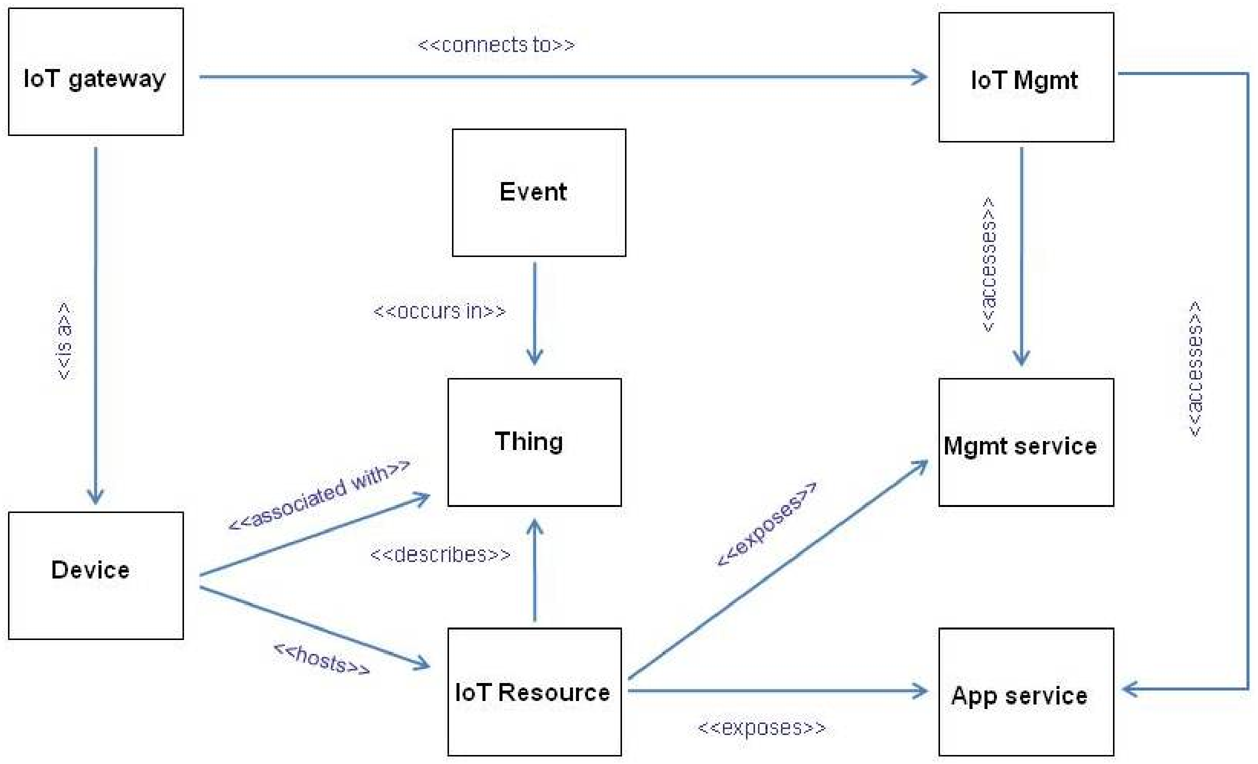
While IoT focuses on delivery of data to and from the devices and the application layer, often data needs to be stored and later accessed by applications and / or by the IoT Services for its proper operation and management of IoT resources. This storage should not follow only structured approach but 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.

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.



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.

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_rREKWt7g
[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).
[OGC SWE]	Open Geospatial Consotium, "Sensor Web Enablement", retrieved at http://www.opengeospatial.org/ogc/markets-technologies/swe (accessed November 14, 2011).
[OpenMeter]	OpenMeter website, retrieved at http://www.openmeter.com/?q=node/8 (accessed November 14, 2011).
[ONVIF]	ONVIF website, retrieved at http://www.onvif.org/ (accessed November 14, 2011).
[SNRA]	ISO/IEC Sensor Network Reference Architecture, draft, retrieved at http://hes-standards.org/doc/SC25_WG1_N1478.pdf (accessed November 14, 2011).

FI-WARE Applications/Services Ecosystem and Delivery Framework

Overview

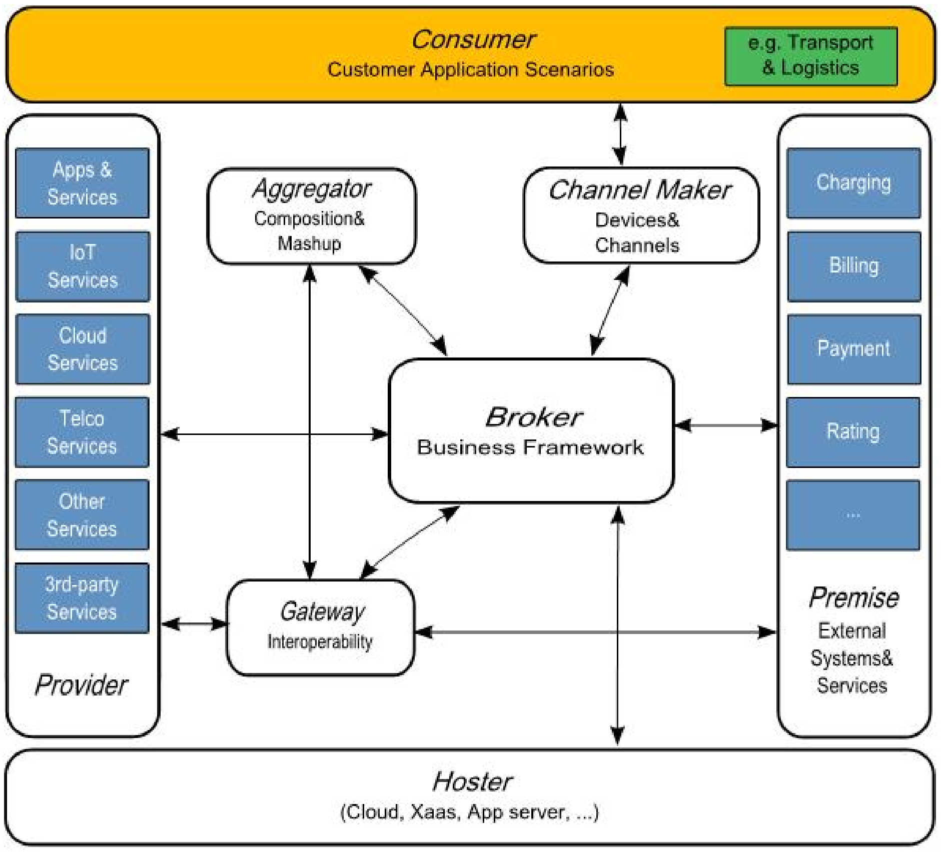
The Application and Services Ecosystem and Delivery Framework in FI-WARE comprises a set of generic enablers (i.e. reusable and commonly shared functional building blocks serving a multiplicity of usage areas across various sectors) for creation, composition, delivery, monetisation, and usage of applications 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.

To avoid misunderstandings we want to emphasize that FI-Ware will not build an ecosystem, FI-Ware will rather provice Generic Enablers for a core business platform, which will offer tradability, monetization, revenue sharing, payment, ..., important ingredients for a business ecosystem, after customization and domain-specific adaptation for USDL and the GEs as well as some complementary components. Also for Composition and Mash-up FI-Ware "only" offers technologies brought in by the partners and not an universal answer to all composition & mash-up problems, which is clearly beyond the scope of the project.

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.



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).

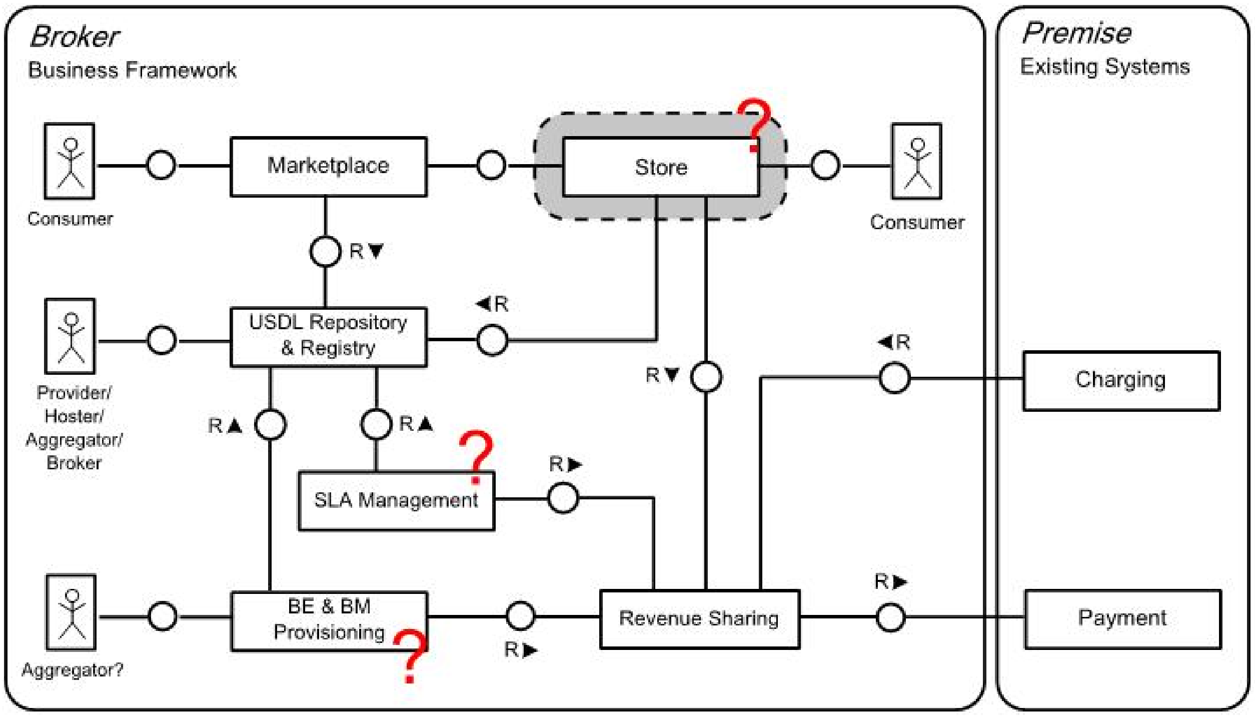
The Consumer supports consumer environments so that the services they require are integrated with the Broker through inbound and outbound run-time interactions. Recall, the Broker allows services to be accessed through different delivery models. Interfaces are required on remote consumer environments 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

Generic Enablers of the Business Framework

The following figure 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.



High-level architecture of the Business Framework*

(*) - The red question marks indicate issues discussed in the question marks section

The set of Generic Enablers (GE) identified are:

- **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.

USDL Service Descriptions

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, RESERVOIR, MASTER, ServFace, SHAPE, SLA@SOI, SOA4ALL), and the Australian Smart Services CRC.)

1- http://fast-fp7project.morfeo-project.org/

2- http://www.reservoir-fp7.eu/

3- http://www.master-fp7.eu/

4- http://141.76.40.158/Servface/

5- http://www.shape-project.eu/

6- http://sla-at-soi.eu/

7- http://www.soa4all.eu/

8- http://www.smartservicescrc.com.au/

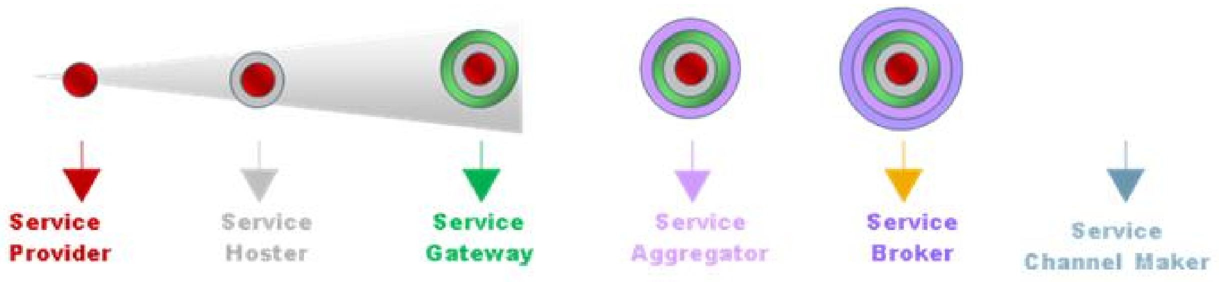
The kinds of services targeted for coverage through USDL include: purely human/professional (e.g. project management and consultancy), transactional (e.g. purchase order requisition), informational (e.g. spatial and demography look-ups), software component (e.g. software widgets for download), digital 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.

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.



Roles of participants in the Internet of Services Ecosystem

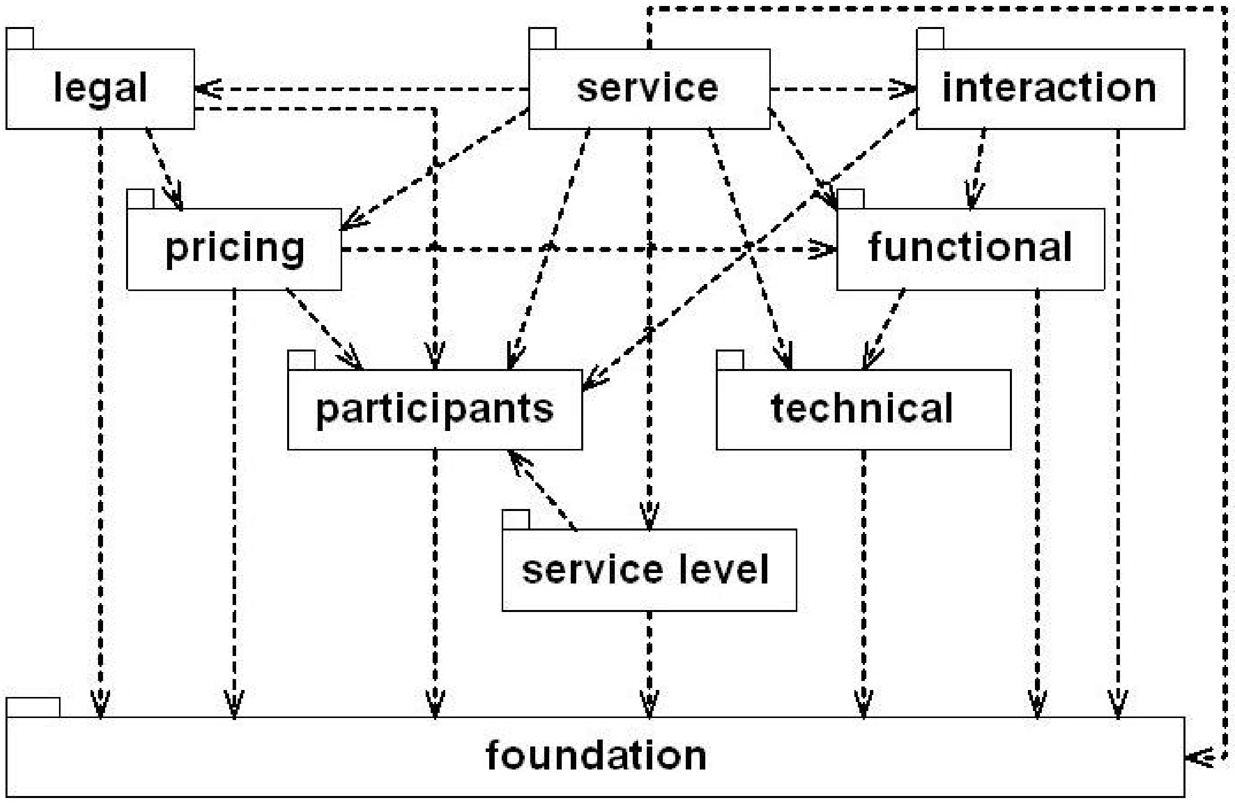
GE description

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.



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.

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 e^{3}Service 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.

Standardization issues

Relations to other standards

USDL is dedicated for business aspects of the services business framework and has no means of describing the technical aspects of services. Therefore USDL is meant to be used in conjunction with other Web standards such as WSDL etc. in order to address detailed technical information. In the case of WSDL, USDL refers to a WSDL description and can even map high-level interactions to elements of a technical service description in WSDL.

The standardization of USDL was discussed in the W3C USDL Incubator Group, which published the final report in September 2011. The common understanding and recommendation of the group members, was to keep the alignment with Web standards and go for a W3C standardization process. So the discussion with the W3C has started to establish a new standardization group in 2012. With respect to the W3C policies and processes, we estimate that the group could be in place in spring 2012.

Repository

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.

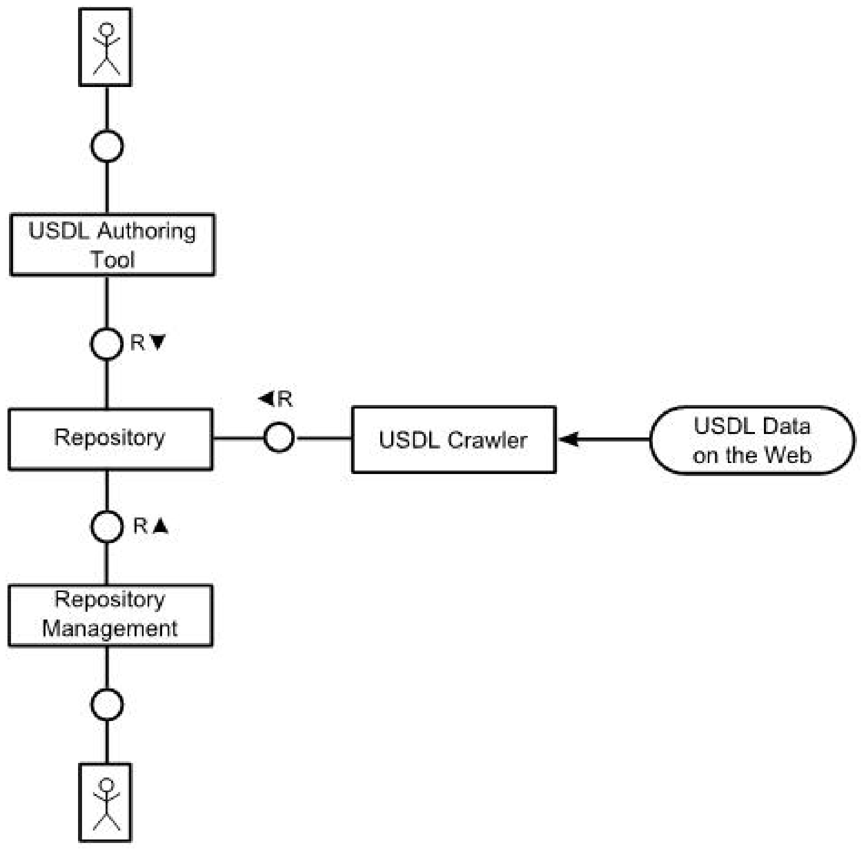
There can be many repositories. A repository can be operated by the service provider (his own web presence), the market place owner as well as any other stakeholder.

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.

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.



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.

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.

Registry

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.

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

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.

Marketplace

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.

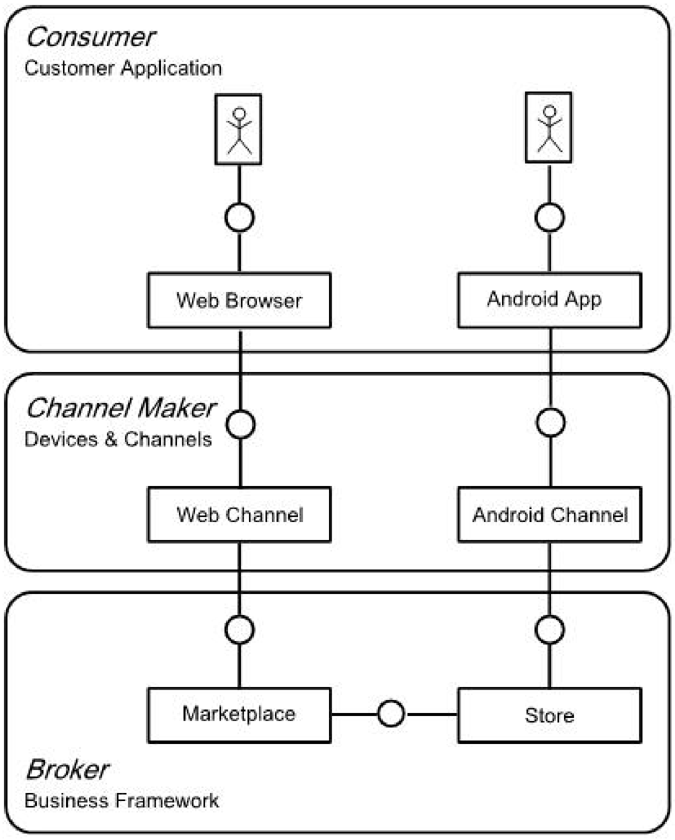
Given the multitude of apps and services that will be available on the Future Internet, providing efficient and seamless capabilities to locate those services and their providers will become key to establish service and app stores. Besides well-known existing commercial application stores like Apple 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

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.

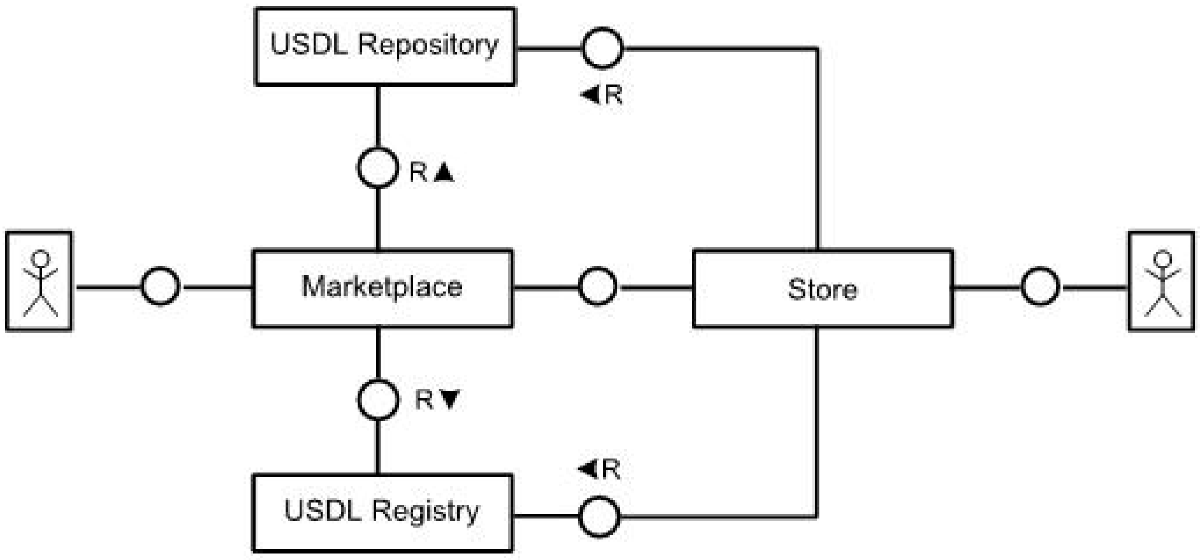


Service Marketplace and Store to Consumer

The figure above 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.

The following figure 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.



Marketplace and Store to Registry

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. The challenge is to provide an interface for markteplace and stores to share descriptions of applications and services without the need to adapt to store specific data formats and APIs.

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.

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 (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.. 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.

Business Models & Elements Provisioning System

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.

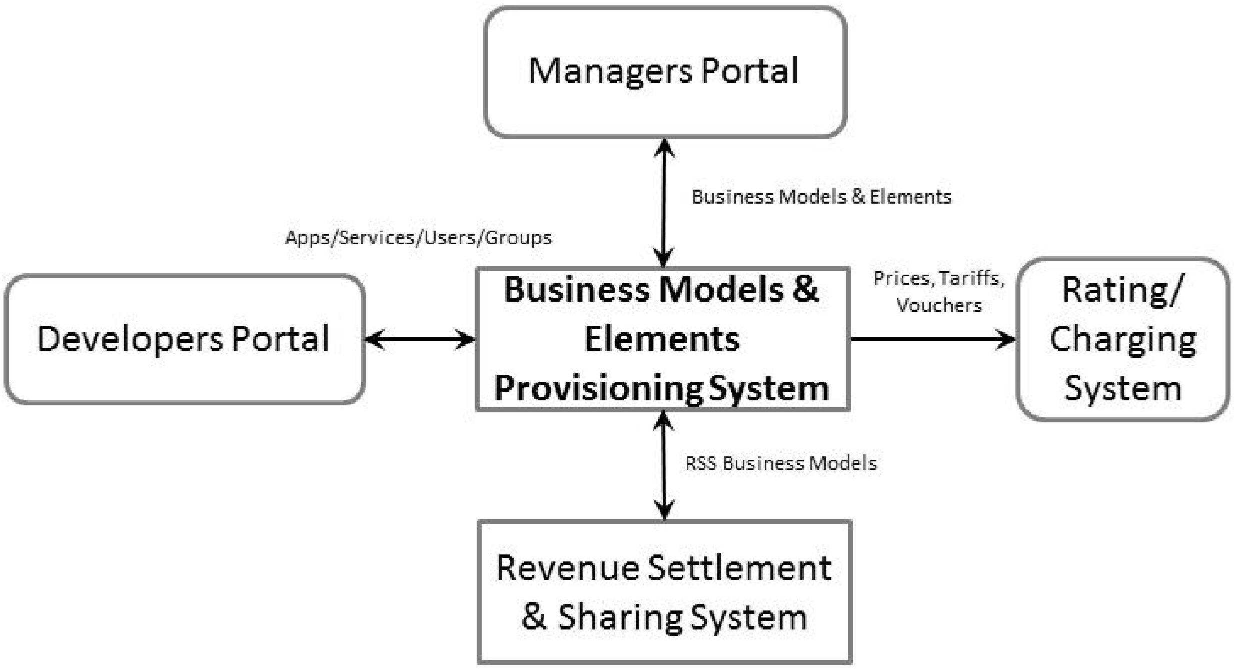
GE description

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
- 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.


Managers Portal
Business Models & Elements
Apps/Services/Users/Groups
Developers Portal
Business Models & Elements Provisioning System
Prices, Tariffs, Vouchers
Rating/ Charging System
RSS Business Models
Revenue Settlement & Sharing System

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

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.

Revenue Settlement & Sharing System

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

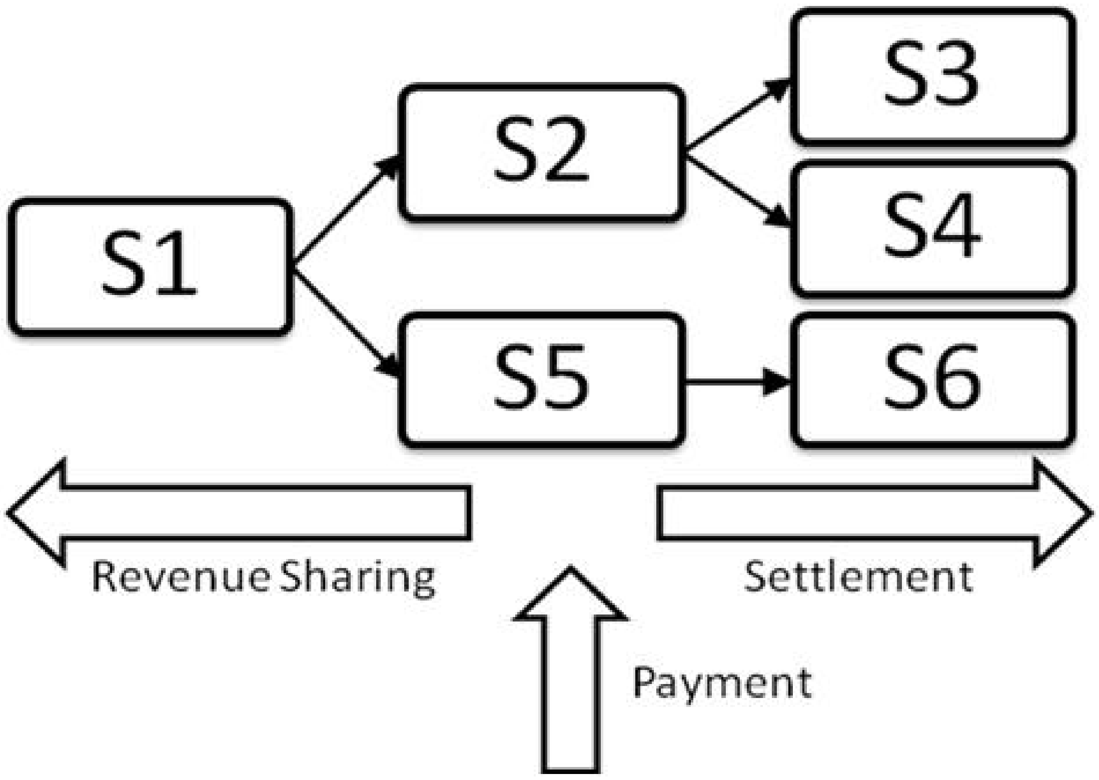
- Managers will setup available business models and parts of them are the revenue share model.
- Service provider will setup revenue share model associated to Applications and services, and they have to be loaded in the Revenue Settlement & Sharing System.
- Developers have to know about the revenues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

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.



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 (1), 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(2) or Orange's Partner(3), 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(4) and Rollbase(5)

1- http://store.apple.com

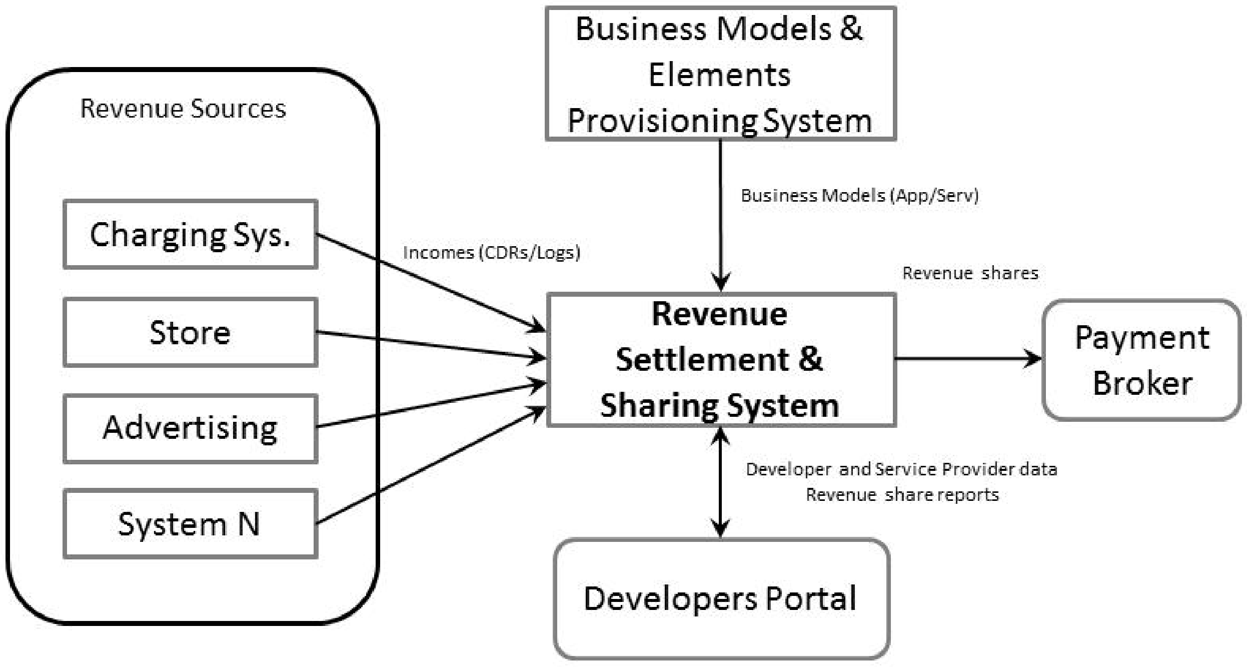
2- http://www.bluevia.com/

3- http://www.orangepartner.com

4- http://www.dbflex.net/

5- http://www.rollbase.com/

The following figure 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.


Business Models & Elements Provisioning System
Revenue Sources
Charging Sys.
Store
Advertising
System N
Business Models (App/Serv)
Incomes (CDRs/Logs)
Revenue shares
Revenue Settlement & Sharing System
Payment Broker
Developer and Service Provider data
Revenue share reports
Developers Portal

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

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.

SLA Management

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:

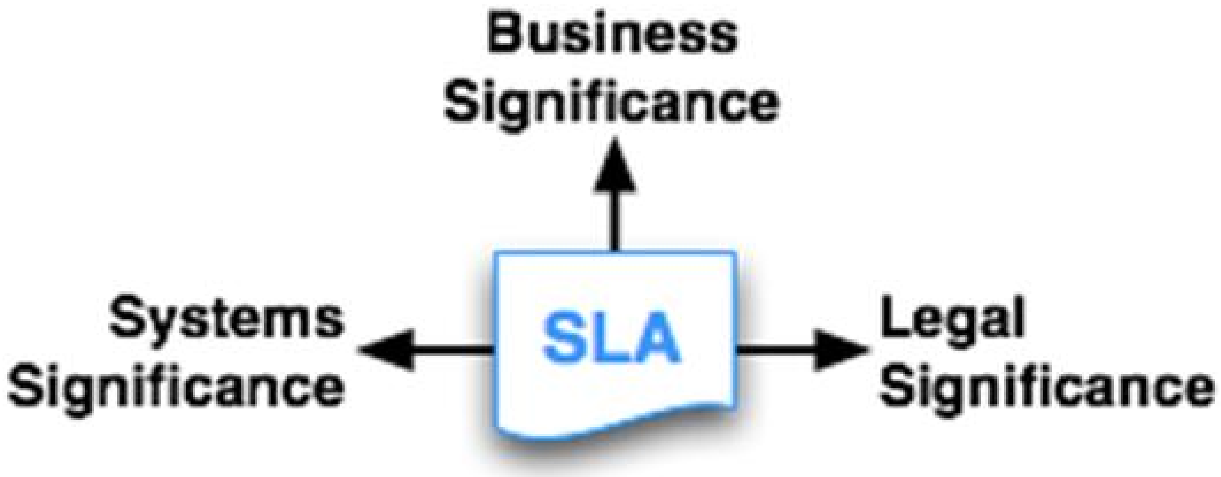
- Service providers design & publish SLA templates as rich descriptions of their service offers.
- Consumers search SLA template repositories for service offers matching their functional & non-functional (QoS, economic, security) requirements.
- Consumers and providers negotiate SLAs.
- Brokers/Hosters (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.

GE description

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

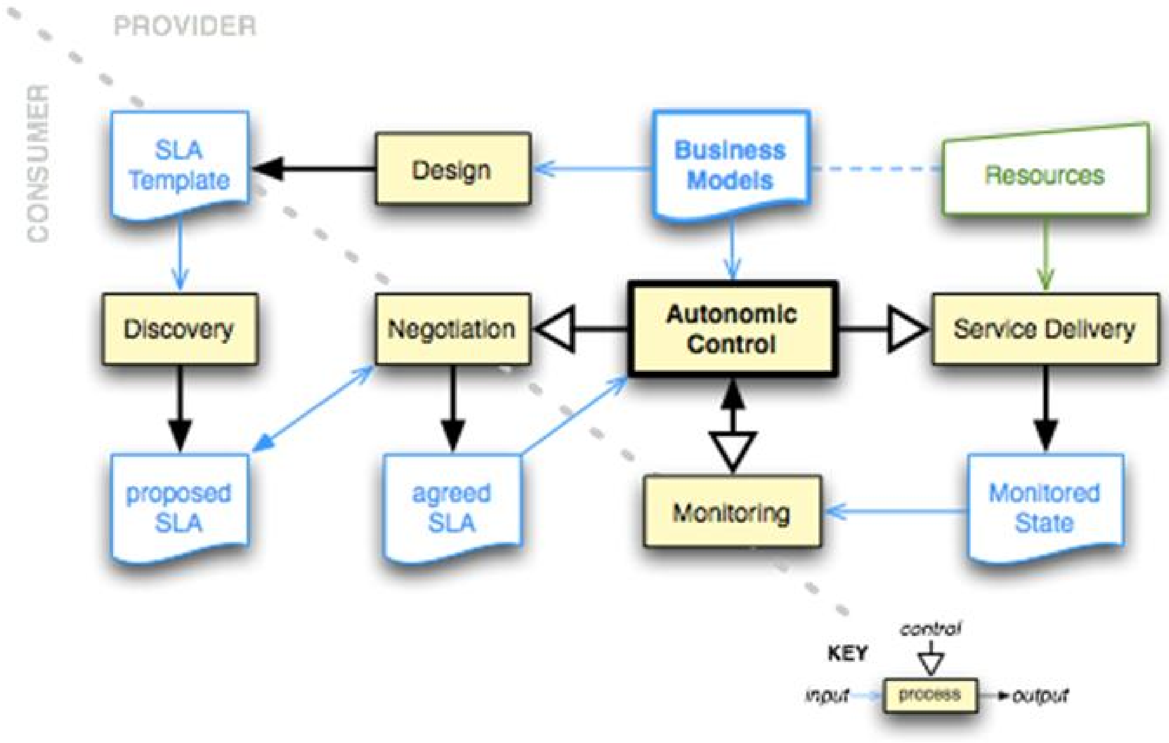
- *Legal Impact*: an SLA represents a binding legal agreement. By entering an agreement, the agreement parties commit themselves to satisfying the terms of the agreement and fulfilling its obligations. To have any significance at all, these obligations must be enforceable – by one means or another - 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.



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.

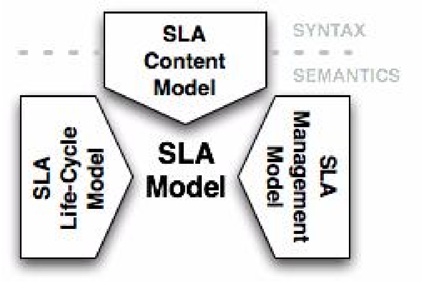


Process view of the SLA life-cycle

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.



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(1), and to consolidate these results with other notable efforts such as: WS-Agreement(2) (*re:* negotiation), WSLA(3) (*re:* content & monitoring) and SLAng(4) (*re*: monitoring).

1- SLA@SOI, see: http://sla-at-soi.eu

2- WS-Agreement, see: http://forge.gridforum.org/projects/graap-wg

3- WSLA, see: http://www.research.ibm.com/wsla/

4- SLAng, see: http://uclslang.sourceforge.net/index.php

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.

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.

Existing products

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

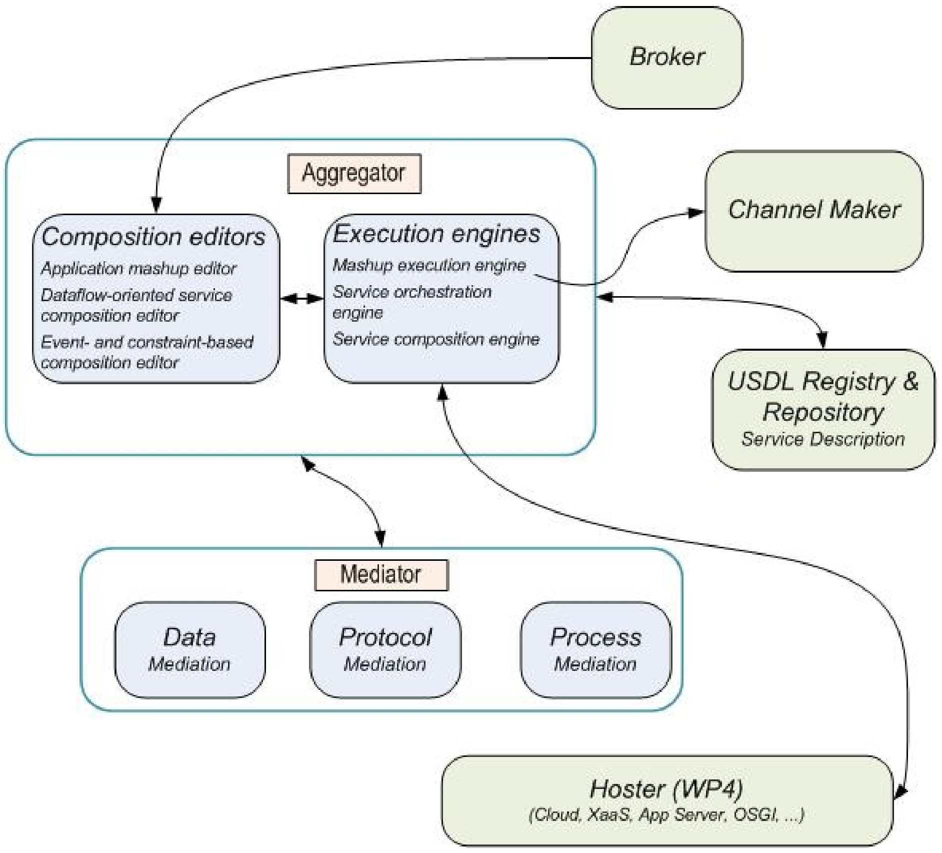
Generic Enablers for Composition and Mashup

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.


Broker
Aggregator
Composition editors
Application mashup editor
Dataflow-oriented service composition editor
Event- and constraint-based composition editor
Execution engines
Mashup execution engine
Service orchestration engine
Service composition engine
Channel Maker
USDL Registry & Repository
Service Description
Mediator
Data
Mediation
Protocol
Mediation
Process
Mediation
Hoster (WP4)
(Cloud, XaaS, App Server, OSGI, ...)

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.

Composition editors

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.

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.

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.

Composition execution engines

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.

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.

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.

Generic Enablers for Gateway

Mediation

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.

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

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.

Generic Enablers for Multi-channel and Multi-device Access

Multi-channel/Multi-device Access System

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

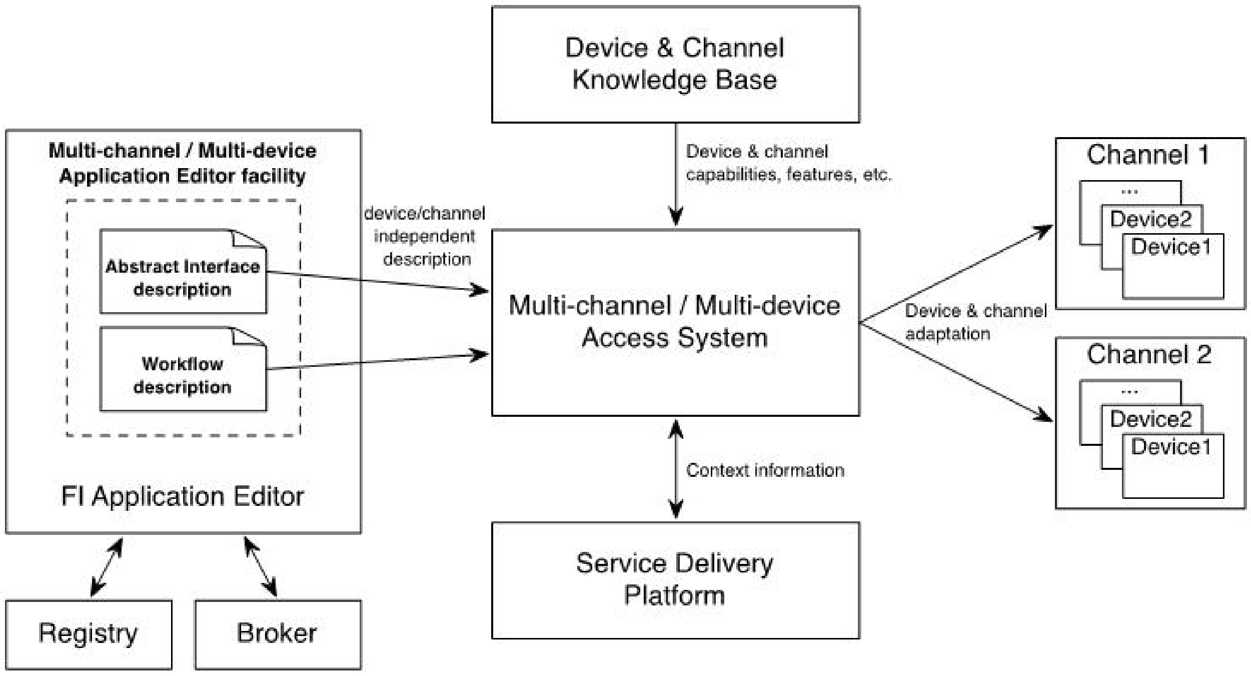
- Application Developer will provide channel specific user interfaces, including the corresponding workflow definition.
- Managers will provide both the content of the knowledge base and access mechanisms to get data from it.

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.


Device & Channel Knowledge Base
Device & channel capabilities, features, etc.
Multi-channel / Multi-device Application Editor facility
Abstract Interface description
Workflow description
FI Application Editor
device/channel independent description
Multi-channel / Multi-device Access System
Device & channel adaptation
Channel 1
...
Device2
Device1
Channel 2
...
Device2
Device1
Context information
Service Delivery Platform
Registry
Broker

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

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.

Question Marks

Relationship to other generic chapters

The six generic domains Security, Internet-of-things service enablement, Interface to network and devices, Data/context management, Cloud hosting, and Applications/Services and delivery framework should also offer their functionality through the business framework and composition & mash-up platform. Indeed, during various meetings this question was discussed. It seems to be a desirable option for exposing the functionality of many GEs to offer it as a service, described in USDL and the business elements and models in order to be sold on the marketplace/store. However, this cannot be the task of the Apps Chapter alone. The Apps chapter will support to enable all kind of GE in FI-Ware for the business framework. It is the a task of the WP2 to foster this kind of collaboration and coherence between the different chapters. In the sense of "eat your own dog food" this would be an interesting proposition for FI-Ware. Nevertheless, we should focus on the Use Case projects, which are the real users of the FI-Ware platform and exploit internal synergies as much as possible, without neglecting the Use Case projects.

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

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.

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

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.
- **Application:** Applications in FI-Ware are composite services that have a IT supported interaction interface (user interface). In most cases consumers do not buy the application they rather buy the right to use the application (user license).
- **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:** We use the term service in a very general sense. A service is a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks. Services could be supported by IT. In this case we say that the interaction with the service provider is through a technical interface (for instance a mobile app user interface or a Web service). Applications could be seen as such IT supported Services that often ar also composite services.
- **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.

FI-WARE Security

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 rendez-vous 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 secured 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 seen by the end-users may be composed of multiple services emanating from many different providers, and the end user can be bewildered 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.

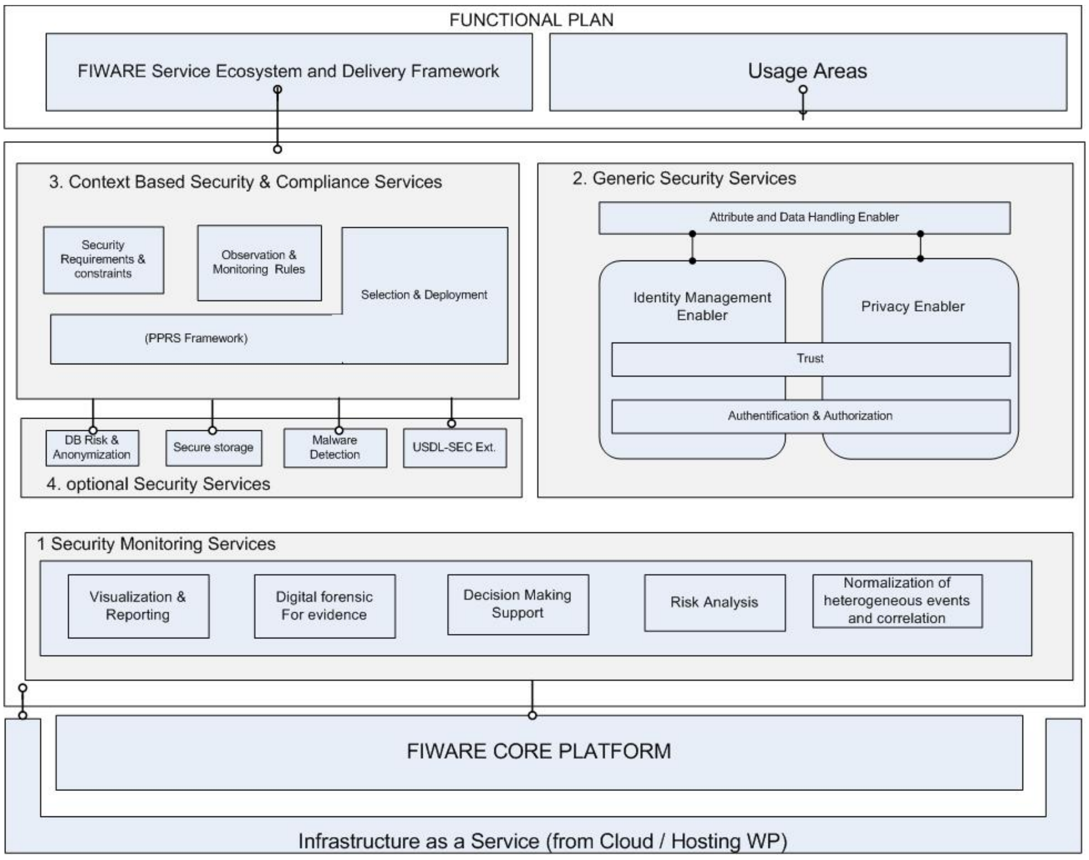
The high level architecture is formed by four main modules: Security monitoring mechanisms (M1), a set of General Core Security Mechanisms (e.g. Identity Management and Privacy solutions) (M2), Context-Based Security and Compliance (M3) where an enhanced version of USDL for security will support the 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 below. 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.


FUNCTIONAL PLAN
FIWARE Service Ecosystem and Delivery Framework
Usage Areas
3. Context Based Security & Compliance Services
Security Requirements & constraints
Observation & Monitoring Rules
Selection & Deployment
(PPRS Framework)
2. Generic Security Services
Attribute and Data Handling Enabler
Identity Management Enabler
Privacy Enabler
Trust
Authentification & Authorization
DB Risk & Anonymization
Secure storage
Malware Detection
USDL-SEC Ext.
4. optional Security Services
1 Security Monitoring Services
Visualization & Reporting
Digital forensic For evidence
Decision Making Support
Risk Analysis
Normalization of heterogeneous events and correlation
FIWARE CORE PLATFORM
Infrastructure as a Service (from Cloud / Hosting WP)

FI-WARE High Level Security Architecture

Security monitoring

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.

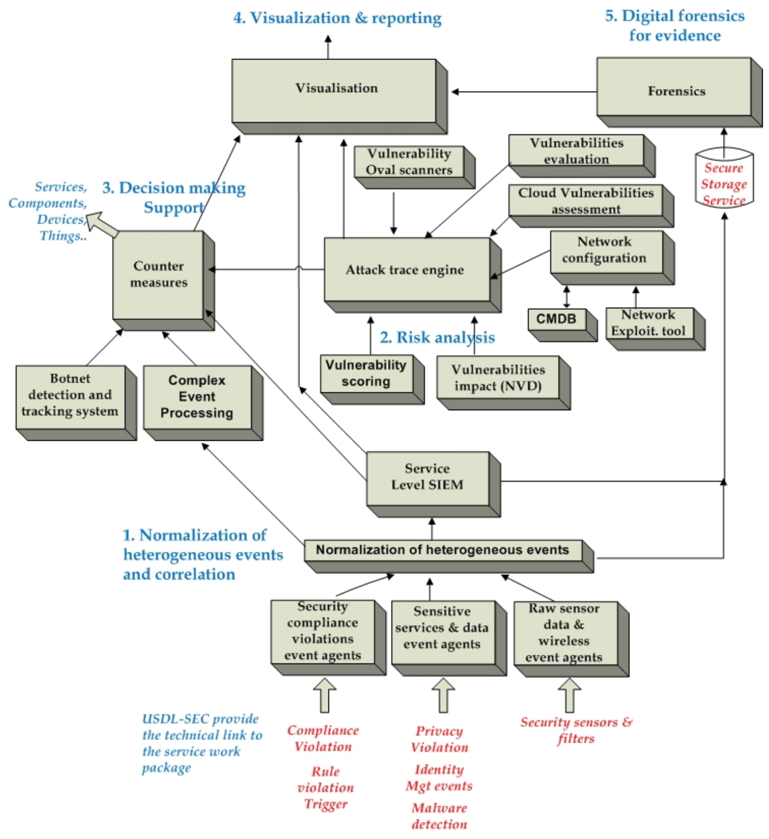
Security monitoring is focused essentially on monitoring alarms from network equipment, systems and security sensors. By the collection, filtering and correlation of data from large-scale heterogeneous environments, including sensitive data from security tools and devices, SCADA events, raw sensor data, 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.

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.


4. Visualization & reporting
5. Digital forensics for evidence
Visualisation
Forensics
Vulnerability Oval scanners
Vulnerabilities evaluation
Cloud Vulnerabilities assessment
Secure Storage Service
Services, Components, Devices, Things..
3. Decision making Support
Counter measures
Attack trace engine
Network configuration
CMDB
Network Exploit. tool
2. Risk analysis
Vulnerability scoring
Vulnerabilities impact (NVD)
Botnet detection and tracking system
Complex Event Processing
Service Level SIEM
1. Normalization of heterogeneous events and correlation
Normalization of heterogeneous events
Security compliance violations event agents
Sensitive services & data event agents
Raw sensor data & wireless event agents
USDL-SEC provide the technical link to the service work package
Compliance Violation
Rule violation Trigger
Privacy Violation
Identity Mgt events
Malware detection
Security sensors & filters

Security Monitoring GE

Normalization of heterogeneous events and correlation

This component is shown as [1] in Figure above. 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).

Security Information and Event management

Security Information and Event Management is a technology that provides real-time analysis of security alerts generated by the component of the FI-WARE architecture. It aggregate data from many sources, providing the ability to consolidate monitored data to notify immediates issues and to help avoid missing crucial events.

Risk analysis

This component is shown as [2] in Figure above. It will cover the following functionalities:

- Vulnerabilities collection and normalization

  The Vulnerability OVAL scanner is designed to evaluate and manage the vulnerability level of Information Systems. Besides producing a list of discovered vulnerabilities, the OVAL scanner can also output a detailed machine configuration information

  The Open Vulnerabilty and Assessment Language (OVAL) 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.

- Vulnerabilities impact

  ICAT, a vulnerability database developed by the US National Institute of Standards and Technology, will provide the information about a vulnerability's effect. The effect classification of a vulnerability indicates how it can be exploited and what is the consequence. The associate vulnerability enables a attacker to execute arbitrary code with all the program's privileges. Fortunately, the ICAT database classifies the effect of a vulnerability in two dimensions: exploitable range (local, remote) and consequences (confidentiality loss, integrity). A local exploit requires that the attacker already have some local access on the host. A remote exploit does not have this requirement. Two most common exploit consequences are privilege escalation and denial of service.

- Network configuration

  The model of network is an abstract host access-control list provided by a network exploitation tool and also by entries from Cloud hosting configuration management data base.

- Attack trace engine

  A series of successfully exploits is a set of attack trace. The objective is to preemptively identify the attack trace and to avoid the access to the assets being the target of an attack. The discovery of these traces requires a reasoning engine analyzing the inputs from vulnerability Oval scanners, network configuration, vulnerabilities impact and Common Vulnerability Scoring System (CVSS-SIG).

- Service risk level evaluation
  Of course, it's not easy to evaluate the formal service risk level, but it is possible to identify critical services and sensitive data and to establish some priorities and adapted countermeasures.

  The goal is:

  - to detect risks affecting the realization of company objectives and non-compliance with regulatory requirements.

  - to 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,

  - to determine whether the level of risk is acceptable or requires mitigation,

  - to adopt measures preventing situations that can compromise security and trust of services.

Decision making support and countermeasures

The Security Monitoring GE support manual or semi-automated selection of countermeasures (shown as [3] in Figure above) mitigating cyber-attacks and reduces the service risk level. This functionality is capable of proposing the most appropriate mitigation responses based on results of the risk analysis functionality.

Countermeasures can involve service 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 above) 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-perspective user interface, able to present a high-level service view, illustrating the impact on the current service transaction, 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 above) 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.

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,
- 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)
- Service-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.

Generic Enablers

Generic Security Enablers Architecture

The Generic Security Enablers consists of three primary sets of application level functionality based around the areas of identity management, privacy and data handling. Together they provide important elements of the trusted environment that is required for users, services and IoT to operate.

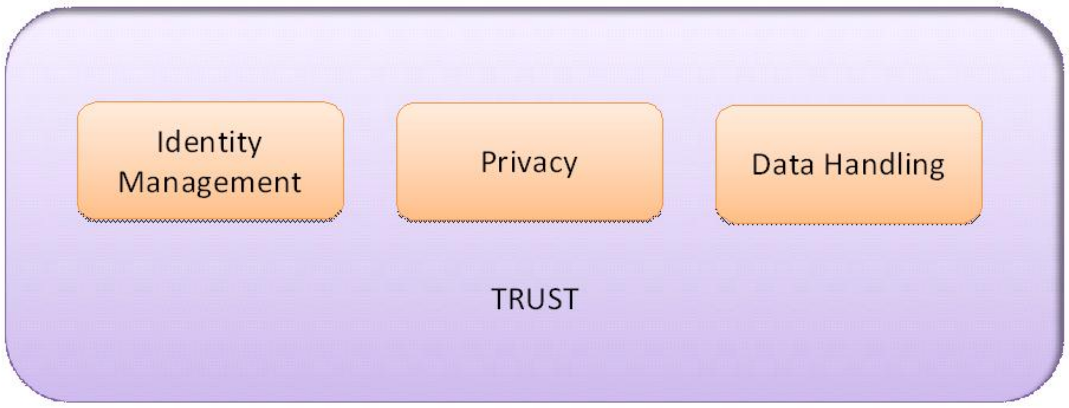


Figure 1: Generic Security Enablers – Overview

Component and Functionality Overview

The generic security enablers will provide a series of application level services to other components of the FI-PPP community. These external interfaces will be standardized and implemented using a set of security assets described later in this document. The use of standardized interfaces will allow the incorporation of multiple technologies in the security architecture. The following diagram shows a high level overview of the three generic enablers and the interfaces that they expose.

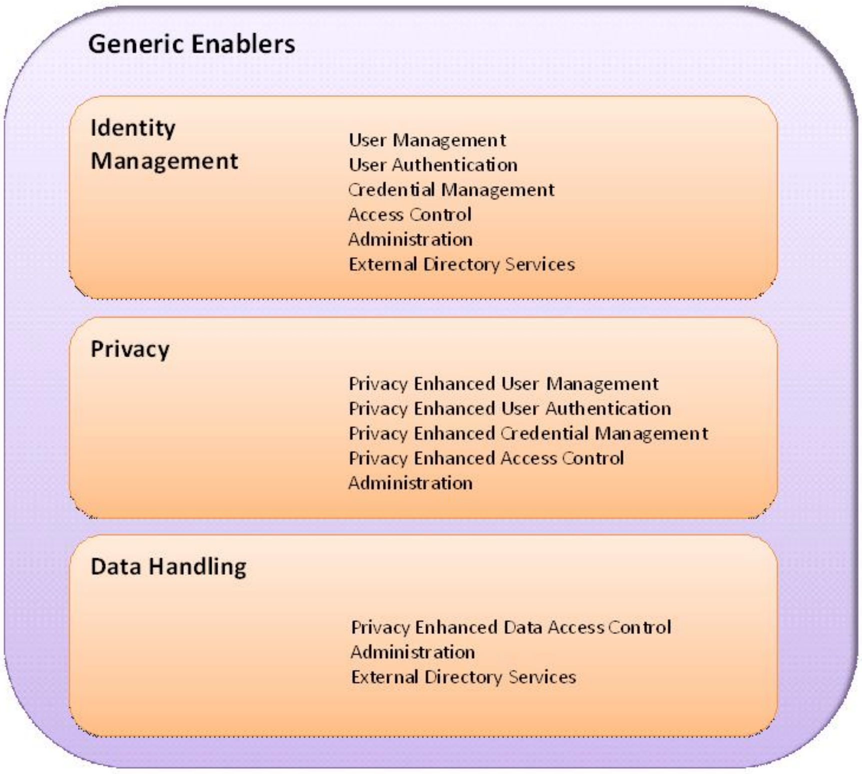

Generic Enablers
Identity Management
User Management
User Authentication
Credential Management
Access Control
Administration
External Directory Services
Privacy
Privacy Enhanced User Management
Privacy Enhanced User Authentication
Privacy Enhanced Credential Management
Privacy Enhanced Access Control
Administration
Data Handling
Privacy Enhanced Data Access Control
Administration
External Directory Services

Figure 2: High Level Functionalities of the Generic Security Enabler

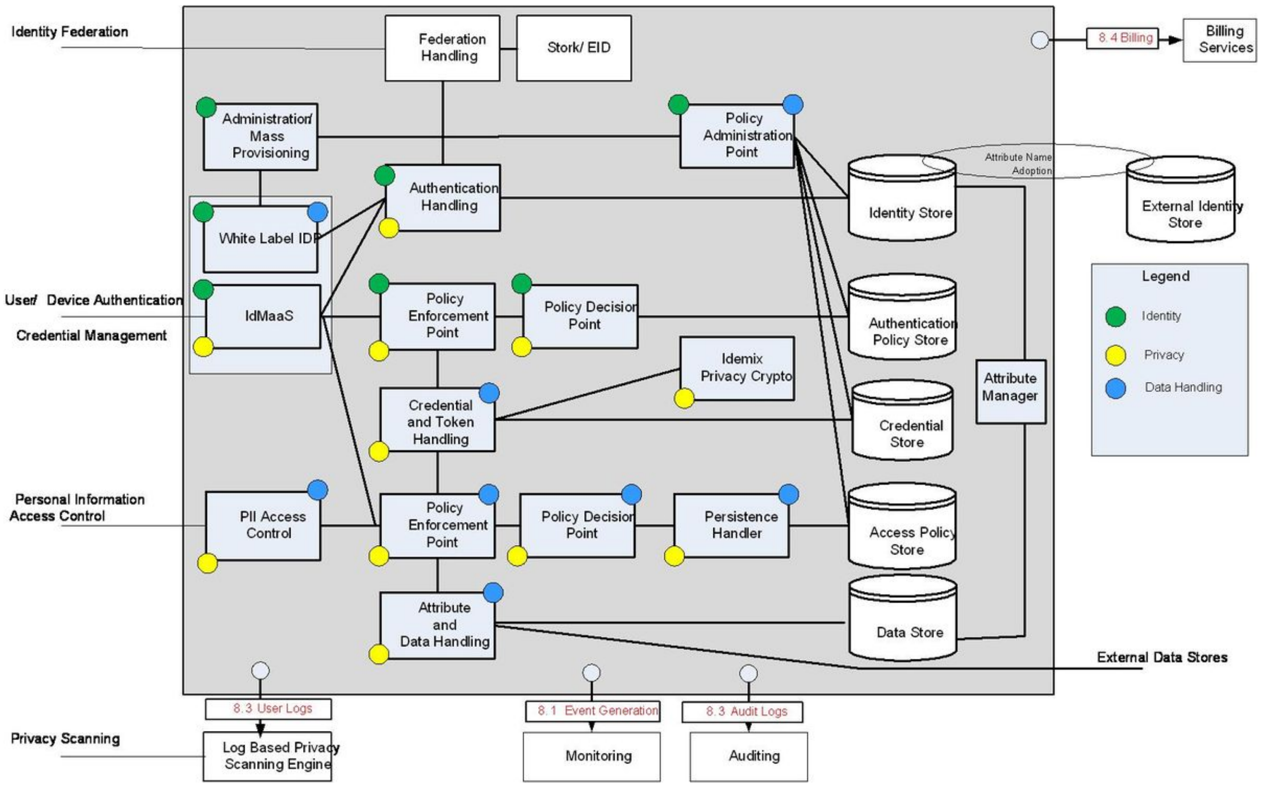


Figure 0: Functional Architecture of the Generic Security Enabler

Identity Management Generic Enabler

Target usage

This enabler provides authentication/access control and identity/attribute assertions as a service to relying parties. The relying parties are typically service providers that provide easy and secure access to their services to users/IoT/other services for instance by means of SSO and that rely on (personal user) attributes (e.g. preferences, location, home address, etc). The users need easy access (SSO) to the growing number of services, and many of them also prefer their personal/identity attributes to be maintained by a trusted party 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 OAuth [OAuth] 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).

Component Description

This component provides identity management functionality that can be used to manage the life cycle of Users and Roles. This includes functionality to register a user, update information associated with a user and eventually to delete a user. Further functionality includes the capability for a user to be set into a state where they are not deleted from the system but marked with a special status that indicates they have been disabled. The IDM solution consists of the IdM “front end” (IdMaaS) and the IdM “back end” (Authentication). The IdM solution 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 Policy Support consists of Policy Enforcement Points and the Policy Decision Point which control the access to attribute which are retrieved from the Policy Store. The Identity Store is a database of identities including identifiers and other attributes. It can be complemented with external stores as well.

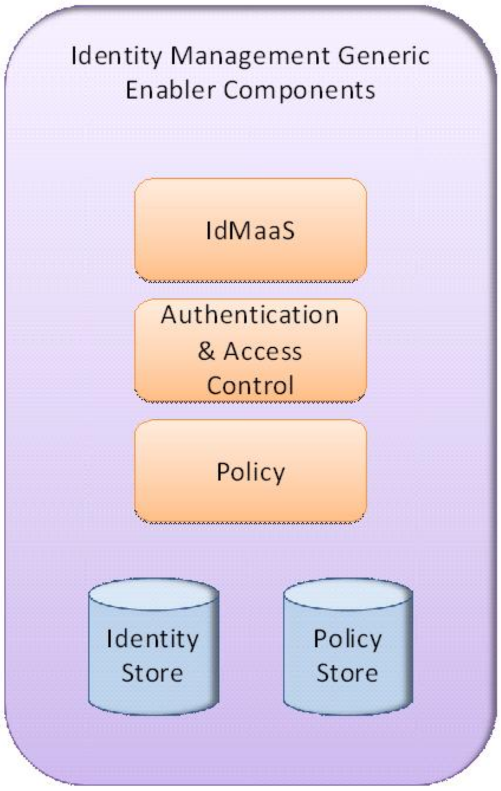

Identity Management Generic Enabler Components
IdMaaS
Authentication & Access Control
Policy
Identity Store
Policy Store

Figure 3: Identity management - High Level Component Overview

Functional Description

This section describes the functionalities provided by the Identity Management generic enabler.

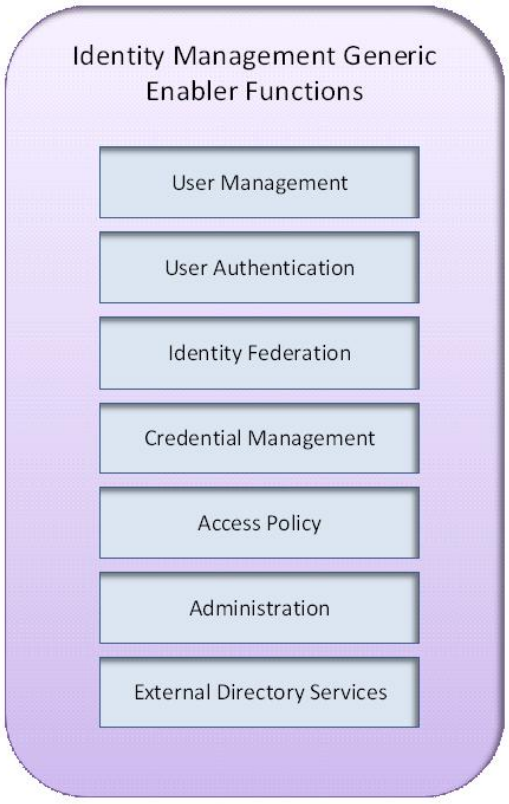


Figure 4: Identity management - Generic Enabler Functions

User Management This functionality is used to manage a user's identity in the system. This is a pre-requisite for providing a trusted relationship between multiple parties. There are several features that are required in order for the life cycle of a user to be managed. These features include e.g.: • User Registration – where a new user is introduced into the system • User Information Update – where information about a user is updated • User Deletion – where a user is removed from the system • User Revocation – where a users rights are removed but they are left in the system

User Authentication Provides a set of functionalities that a service can use to authenticate a user. There are a number of open standards that will be supported which include in a first phase: • Open ID • OAuth • Oasis / SAML • user id and password

Adaptation to specific api's or support of other standard interfaces might be considered on request.

Identity Federation This interface provides Identity federation functionality which allows a single sign on to be provided. There are a number of open standards that will be supported which include: • Liberty Alliance / Kantara

Credential Management Credentials or tokens are things used to convey a Users right to access a resource. This interface provides a series of functions to manage these credentials. Supported credentials include: • X509 certificate • RSA Tokens • HW Dongles • SAML tokens

These credentials need to be provided to a user in order that a service may be accessed. There are a set of features that allow a credential life cycle to be supported. These features include:

• Issue Credential • Renew Credential • Delete Credential • Revoke Credential

Access Policy Access Policies specify the conditions a user/service has to satisfy in order to obtain access to a resource. Such a policy can be expressed and interpreted 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.

Administration This function enables an administrator to configure various aspects of the system and to provision and link user accounts.

External Directory Services An external directory service is an alternative for verifying user's credentials (username and password) in case they are unknown to the IdM Generic Enabler System.

Critical product attributes

- Authentication and attribute assertions about users and things as a service to relying parties via open protocols (OAuth, 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 the rules to reveal their attributes via policies.A

Privacy Generic Enabler

Target usage

The privacy generic enabler provides a set of functionality similar in scope to the Identity management generic enabler described in the previous section but enhanced using special privacy enhancing technologies. These privacy-enhancing technologies are primarily centered on special credentials that contain attributes about the user which can be disclosed for authentication purposes selectively on a need-to-know basis.

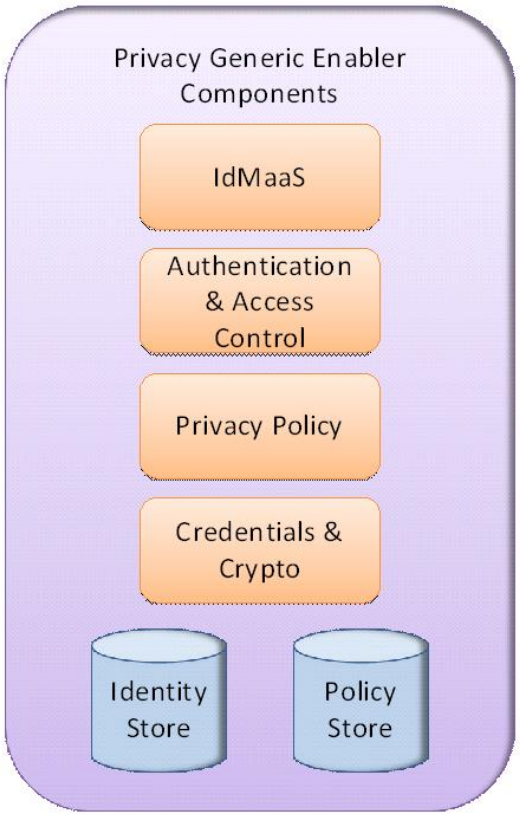


Figure 5: Privacy - High Level Component Overview

Component Description

This component will utilize many of the basic functions provided by the Identity management generic enabler (see Generic Identity Enabler). The Privacy Generic Enabler includes the IDM solution (IdM “front end” (IdMaaS) and the IdM “back end” (Authentication)). It communicates with the relying parties (services, things) as well as with the subjects of authentication and attributes assertions (users, things).

Functional Description

From a functional point of view, the privacy generic enabler is almost identical to the identity management generic enabler. It differs mainly in the technologies that are used to instantiate the several components. In particular, those are the privacy-enhanced access policy and privacy-enhanced credentials that is reflected in the User Authentication and Credential Management.

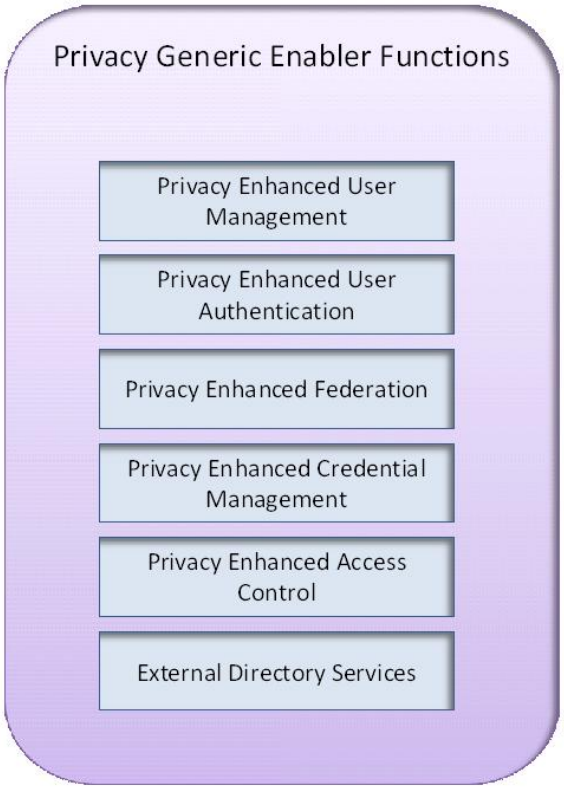

Privacy Generic Enabler Functions
Privacy Enhanced User Management
Privacy Enhanced User Authentication
Privacy Enhanced Federation
Privacy Enhanced Credential Management
Privacy Enhanced Access Control
External Directory Services

Figure 6: Data Usage - High Level Function Overview

Privacy Enhanced User Management See Generic Enabler Identity Management for basic functionality. On top of that this functionality is enhanced by minimum disclosure technology.

Privacy Enhanced User Authentication See Generic Enabler Identity Management for basic functionality. On top of that this functionality is enhanced by minimum disclosure technology.

Privacy Enhanced Federation See Generic Enabler Identity Management.

Privacy Enhanced Credential Management This privacy credential sub element is based on IBM’s Idemix library and provides private credential systems and minimal disclosure tokens for enhanced privacy protection mechanisms. End-users are provided with the possibility of selectively disclosing their asserted private attributes or even just proving predicates over their attributes in an unlinkable manner. At the same time, the cryptographic technologies ensure strong and secure authentication to the service/resource providers 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 attributes 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 privacy-enhanced attribute-based access control, private credentials 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.

Privacy Enhanced Access Control In order to support the full functionality of privacy-enhanced credentials an enriched policy language and token format is needed. The PrimeLife Privacy Policy already extends the standard XACML language to express for the important concept of predicates on attributes. That is, instead of revealing an attribute only the fact if some condition on the attribute is fulfilled will be revealed. Further extensions in the Privacy Enhanced Access Policy will cover the use of unlinkable pseudonyms instead of (identifying) public keys, or the optional lifting of the anonymity. Please refer also to “Data Handling Generic Enabler”.

External Directory Services See Generic Enabler Identity Management.

Critical product attributes

- Unlinkability: Different tokens created by the same user cannot be linked -- unless the token was explicitly created in a linkable way.
- Untraceability: The user who created a token cannot be identified -- unless the token was explicitly to be traceable by a trusted authority.

Data Handling Generic Enabler

Target usage

The component provides a mechanism for controlling the usage of attributes and data (=PII) based on the concept of ‘sticking’ a data usage policy to the data to which it applies. When the data is accessed by an application, an access control technology is then used to verify that the intended use of the data matches the scope defined in the usage policy.

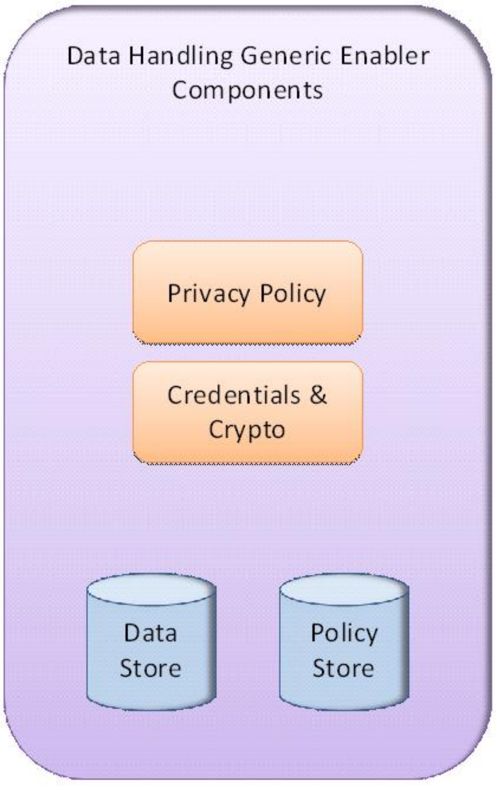


Figure 7: Data Handling - High Level Component Overview

Component Description

Privacy Policy Engine

After the credentials of the user have been verified a check is made which policies apply to the user's request. Policies are stored in the policy store and can be modified by both the users (own policies) and the administrators (all policies), depending on the scope of policy.

Credentials and Crypto

This component is responsible for enciphering the data stored by the user in the Data Store. Enciphering is being done by the public key of the users. In the case the users intend to retrieve the data their private key would be required. On top of that this component authenticates the users by checking the user's credentials, prior to forwarding the requests to the Privacy Policy Engine.

Functional Description

This section describes the external function exposed by the Data Handling generic enabler.

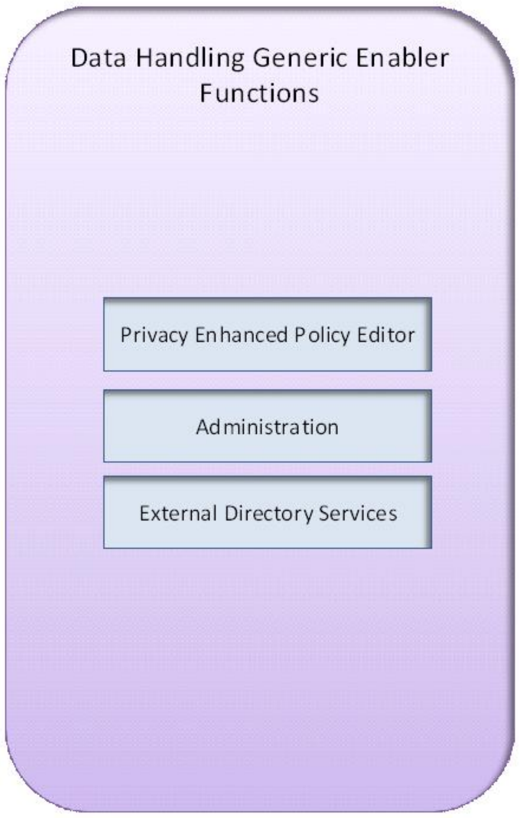

Data Handling Generic Enabler Functions
Privacy Enhanced Policy Editor
Administration
External Directory Services

Figure 8: Data Usage - High Level Function Overview

Privacy Enhanced Policy Editor

This function enables to control access to PII based on policies that have been attached to the data by a user. These data handling policies govern such aspects as to how long data should be retained, if the data may be handed on further to a third party, if it should be deleted after a certain time. If an Attribute Manager is available in the system identity storages can be provided with sticky policies.

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.

User Attribute Handling

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.

Administration

This interface provides functionality for an administrator to configure various aspects of the data handling functionality.

External Directory Services

See Generic Enabler Identity Management.

Critical product attributes

- Provide the users more control about his/her PII stored in the network/cloud.
- Easy en-/decryption of the user’s PII even in unsecured areas provided by the Data Handling Generic Enabler.
- Privacy respecting enforcement of PII with the help of a Privacy Policy Language PPL.

Context-based security and compliance

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.

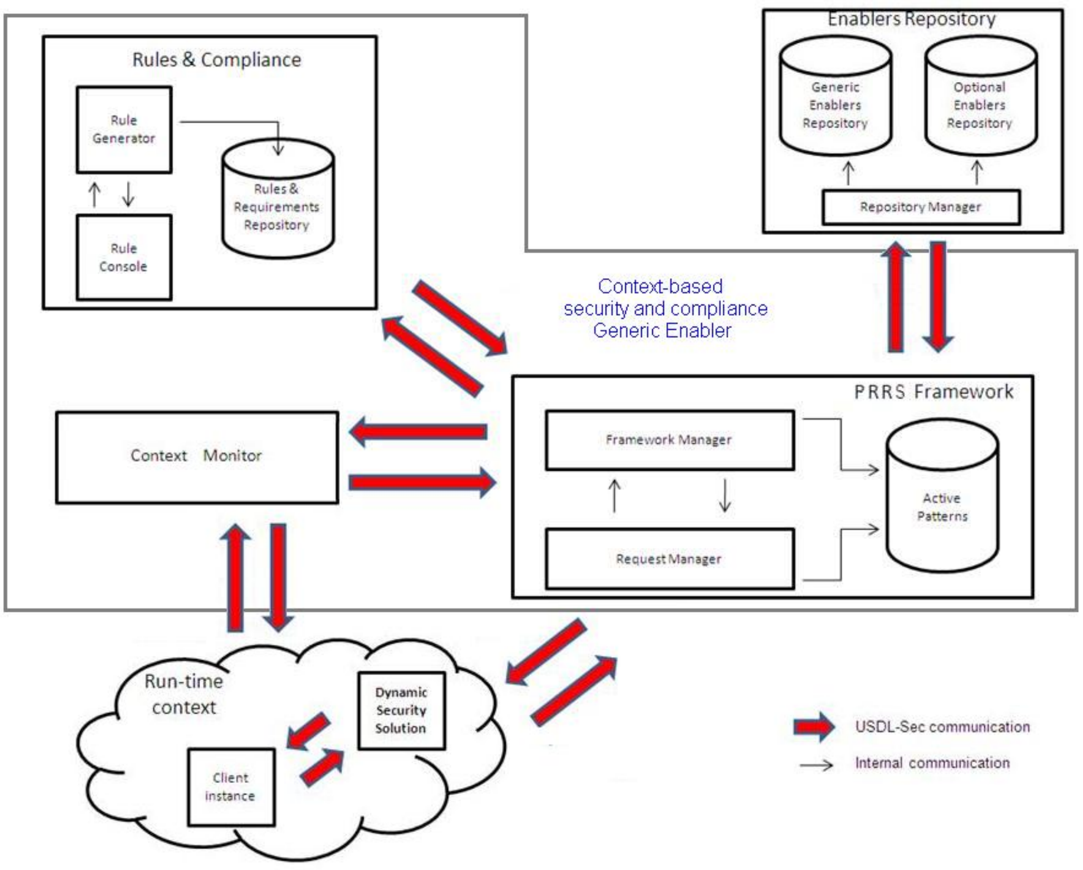
The GE will accept security request from a client application and will select the best Optional Security Enabler to fulfil it.

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.



Context-based Security and Compliance Generic Enabler

GE description

Three main internal components are defined by this Generic Enabler

- PRRS Framework: Provides run-time support to applications performing Dynamic selection & deployment of security enablers.
- Context Monitor: It's able to check the system compliance status and trigger PRRS Framework in case of noncompliance detected.
- Security requirements repository: This component offers the possibility to manage (create, modify, storage & delete) compliance rules to be used.

PRRS Framework

The PRRS framework is the core of the Generic Enabler. It controls the rest of the components of the system; processing requests from end-user applications and orchestrating the instantiation of the Security Enabler selected.

PRRS is implemented as a service running in a device, on top of the operating system, and listening to end-user applications requests.

End-user applications send requests in order to fulfill their security requirements to it either as a specification of security requirements or as a reference to already existing rule.

Once receive a request PRRS will get the applicable rules of compliance from rules' repository and will find the most suitable security enabler from enablers' repository

Then PRRS will deploy the solution as executable components in the context of the end-user application is running

Finally PRRS will provide monitoring devices with the rules that end-user applications context must fulfill and will updated its internal data base

PRSS internal Data Base allows framework to link the deployed solution with the monitor that oversees the user context and the applicable rules the system must fulfill.

After security solution is running properly PRRS framework participation could be requested again either by monitoring system in case a noncompliance is detected or by rules' repository in case of a rule modification is notified.

PRRS will take the necessary recovery actions by the reactivation, reconfiguration, deactivation and/or substitution the deployed enabler.

Anytime PRRS hasn't been able to find a suitable security enabler it would notify the End-user application the situation.

Contex Monitor

Runtime context monitors are the components in charge of detect anomalous behavior or non-conformances in End-user context environments.

Each Monitor component will get context and status events from the end user and the security enablers it is overseeing.

Then it will compare the information obtained with the rules provided by PRRS Framework.

In case of non-compliance detection the assigned event will be send to PRRS framework by the appointed monitor so that the framework could take the necessary recovering actions

Additionally Context monitor will provide a Dashboard that gives the system reporting capabilities.

Reports of system performance will be generated once the information from data context has been compared with rules received from PRRS

These visual reports will provide useful information about the levels of compliance, performance of the solutions dynamically deployed by PRRS and make the task of identification of root-causes for non-compliant situations easy.

Security requirements repository

This component will allow the generic enabler to store and manage compliance requirements and relevant specifics at various abstractions levels and also check end-to-end business processes for compliance against the set of applicable constraints during design-time.

The rules to be stored could come from various sources, including laws and regulations, public and internal policies, standards, customer preferences, partner agreements and jurisdictional provisions.

In order to manage compliance throughout all phases of business process lifecycle, it must be possible to define and subsequently integrate compliance specifics into business processes and enterprise applications, and assure compliance starting from the process analysis and design phase.

Furthermore the component will be is able to reuse fragments of already stored formal rules specifications to built a new formal specification form a new law or rule to be stored

Reuse of rules specification fragments will make the task of compile new laws or rules into formal language easier

Each high-level rule or specification will be compiled into a formal pattern that can be applied and referenced in many scenarios either by end-user applications as a security requirement or any security enabler to describe its characteristics.

Finally Security requirements repository will be able to trigger PRRS framework when some rule will be modified so that the framework could take the necessary actions in case of the modification must be taken into account on compliance measurements

The generic enabler also will describe a new business service oriented language that will be able to describe & register security services, capabilities and compliance rules The communication between end-user applications, the deployed executable components and the framework will be is supported by this service oriented language. The USDL-SEC language will be defined as an extension of existing standard USDL 3.0 by implementing a new security oriented module. USDL-SEC language will support the following basic characteristics:

• Provide mechanisms to cover both high level description of the service and detail functionalities & implementations.

• Describe functional security services provided by the platform and needed by applications with functional and non-functional properties in single and complete description file.

• The Security model will be fully defined by its associated properties file; allowing consumers and providers to agree on a security protocol, through expressions of concrete mechanisms and links to existing standard such as WS-SecurityPolicy, XACML, P3P, etc...

• Provide means to compare and select services according to consumer needs. A service description is first made by a service provider, which then exposes it to a service broker. This will make the PRRS task of selecting a service following user requirements easy.

• Provide event management capabilities to allow monitors get the context event information from the security enablers they overseen.

• Describe requirements and compliance rules to be fulfilled in an specific end-user environment

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.

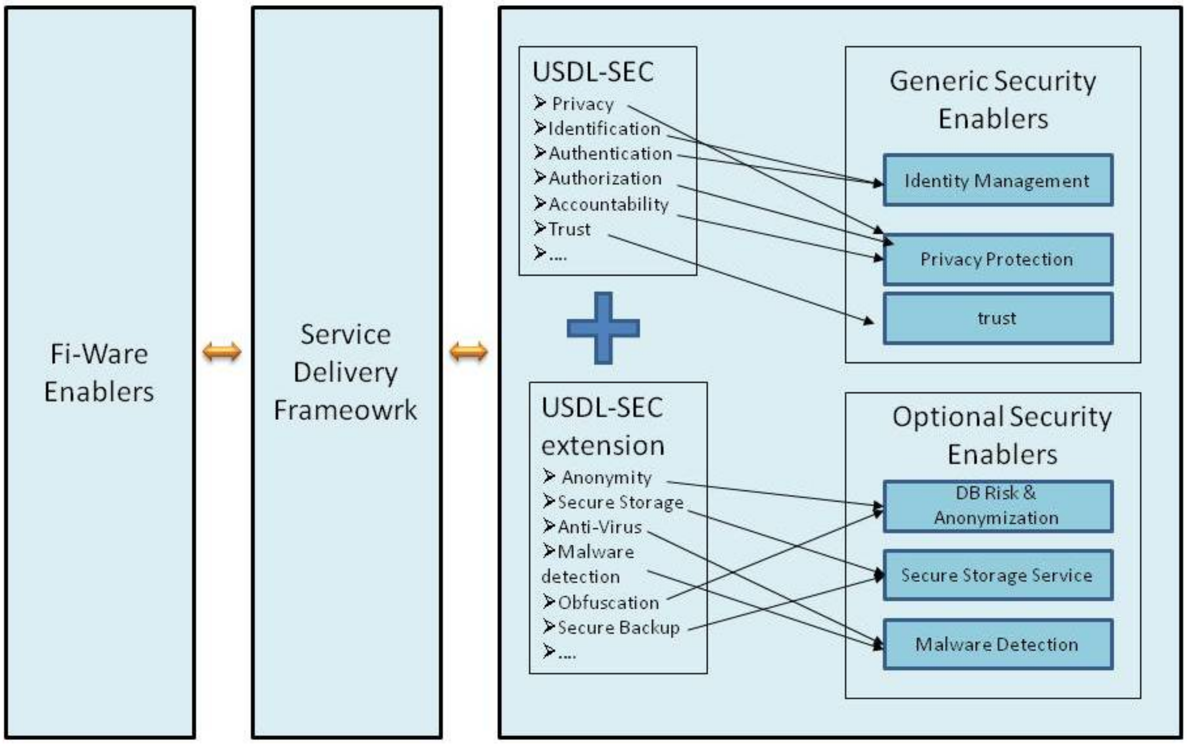
Optional Security Service Enabler

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.



Optional Security Service Enabler

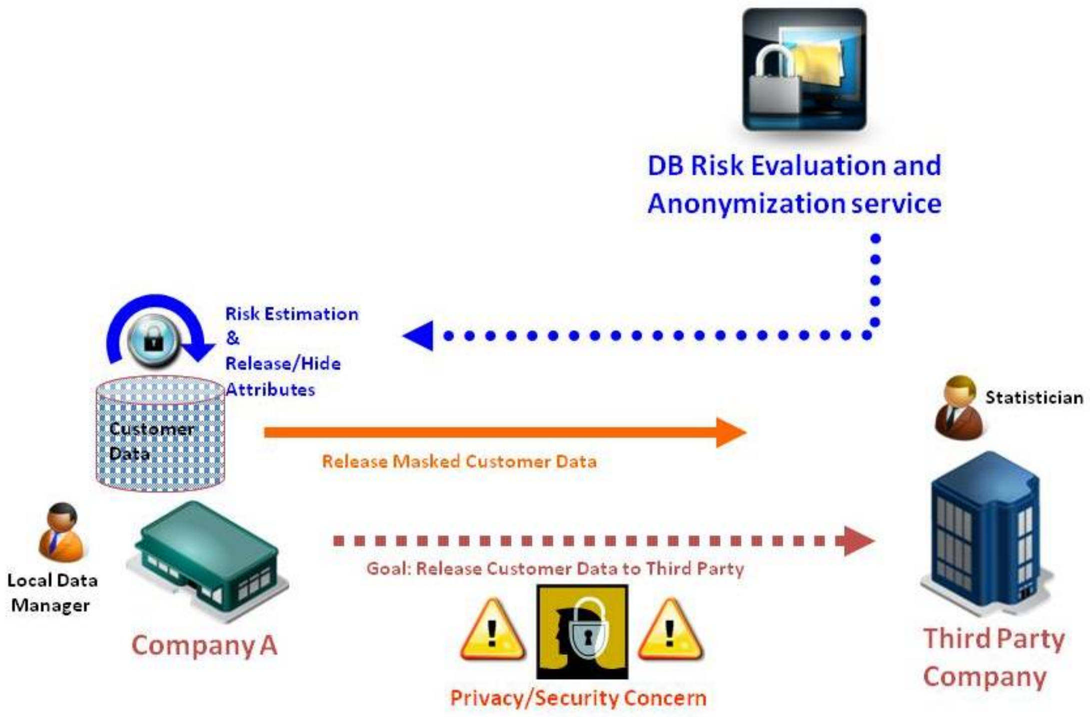
GE description

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.

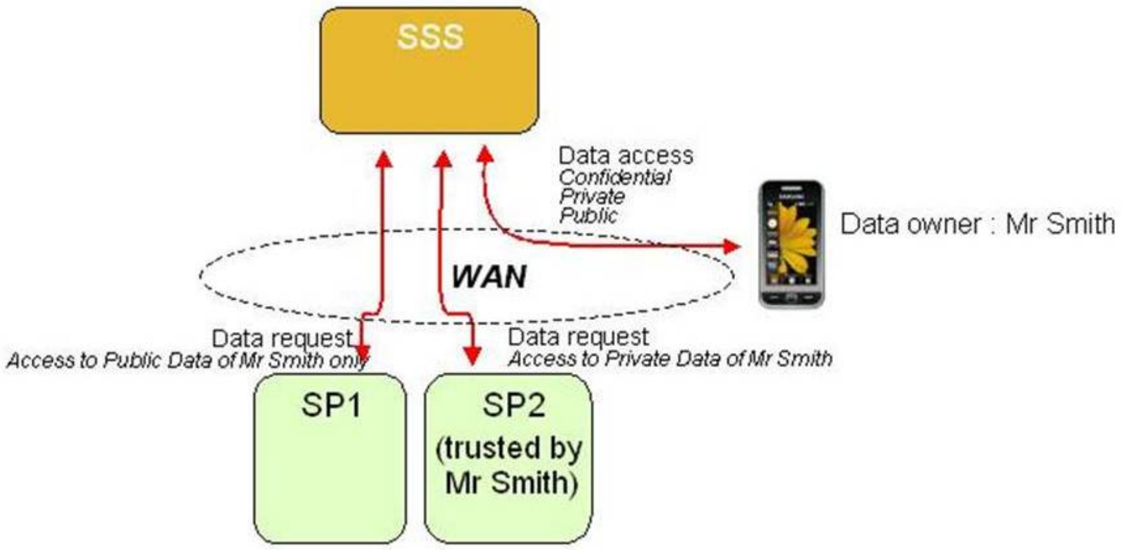
Some example of possible optional security services could be:

- **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.


DB Risk Evaluation and Anonymization service
Risk Estimation & Release/Hide Attributes
Customer Data
Release Masked Customer Data
Statistician
Local Data Manager
Company A
Goal: Release Customer Data to Third Party
Privacy/Security Concern
Third Party Company

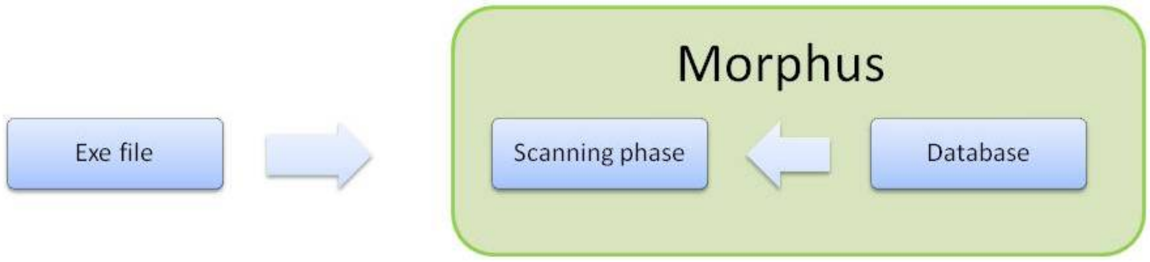
DB Risk Evaluation and Anonymization Service

- **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.


SSS
Data access
Confidential
Private
Public
Data owner : Mr Smith
WAN
Data request
Access to Public Data of Mr Smith only
Data request
Access to Private Data of Mr Smith
SP1
SP2 (trusted by Mr Smith)

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.



Morphus

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

Question Marks

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

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 traces of attack. The problem is that it is not possible, in this case, to identify such traces.

Service risk impact assessment

The difficulty to identify the attack trace in a volatile environment also can be applied to the service risk impact assessment. It is not possible to do it if the resources used by services 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.

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
- **Service impact analysis.** An analysis of a service’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.

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 [1] [1]
[idemixlib]	[2] [2]
[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 [3] [3]

References

[1] http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/IdentityMixer_ProtocolSpecification_2-3-3.pdf
[2] http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/index.html
[3] http://www.internet-of-services.com/

FI-WARE Interface to Networks and Devices (I2ND)

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”.

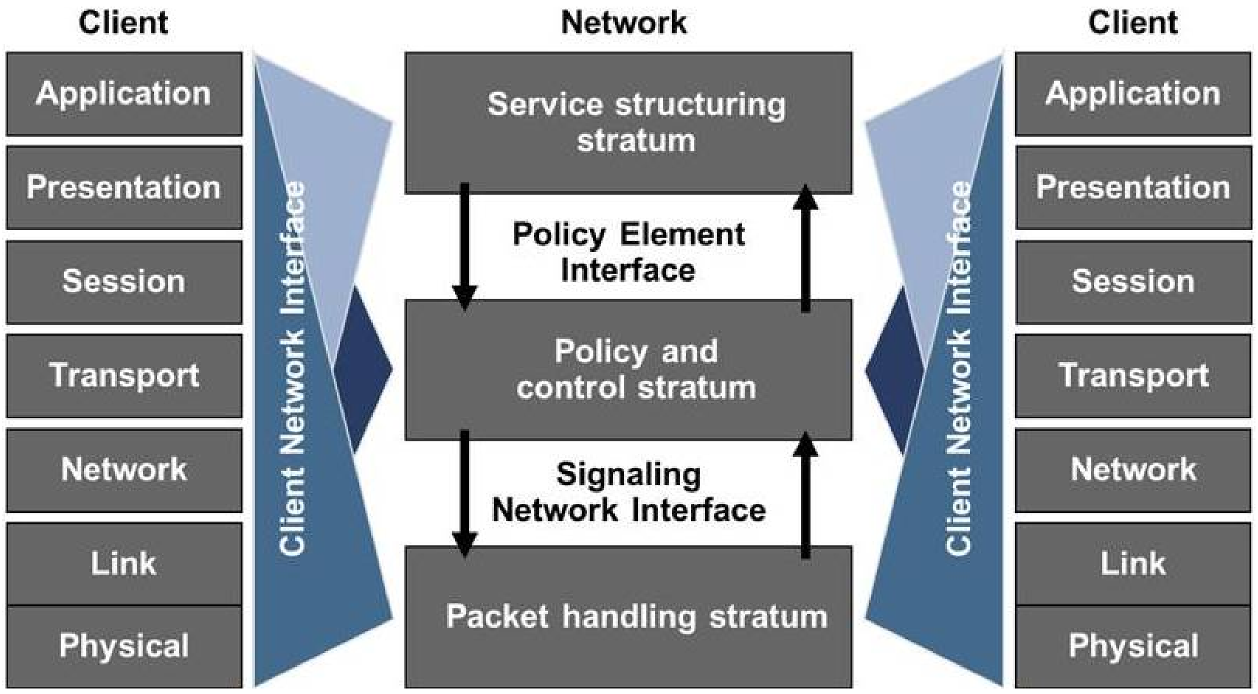
FI-WARE considers the **Intelligent Connectivity** at the basis of the Future Internet. The Intelligent Connectivity concept will mean connecting applications and application platforms to the network intelligently, leveraging on the full potential of the features of the network, through the definition 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.



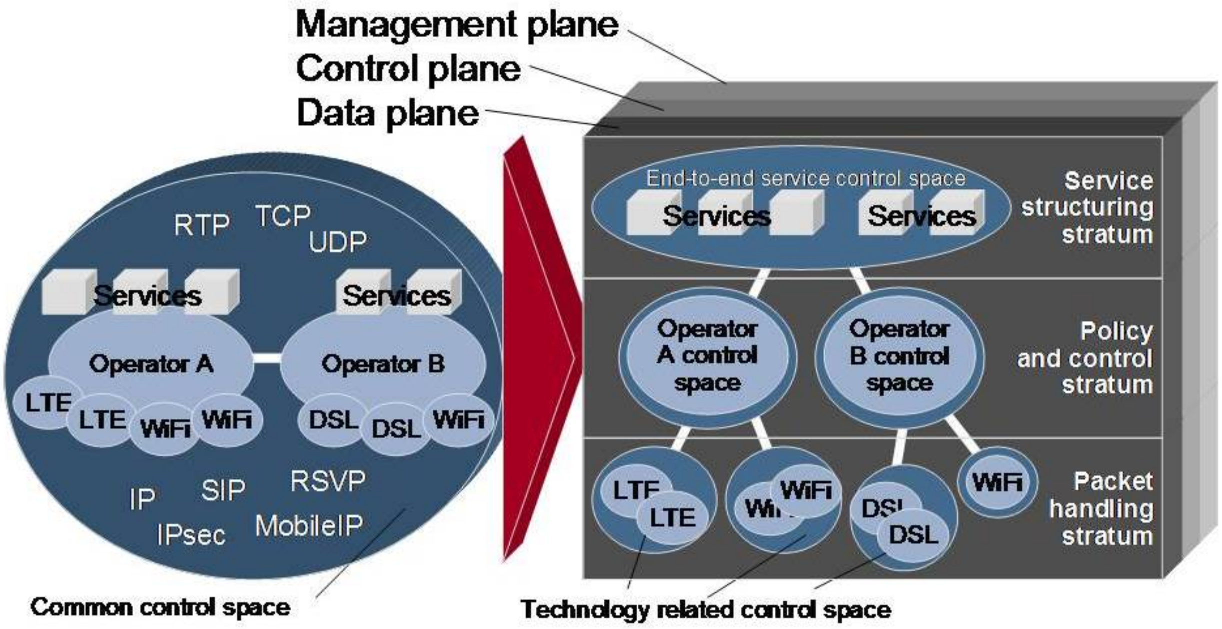
Three strata model for communication in an IP driven operator infrastructure

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



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.

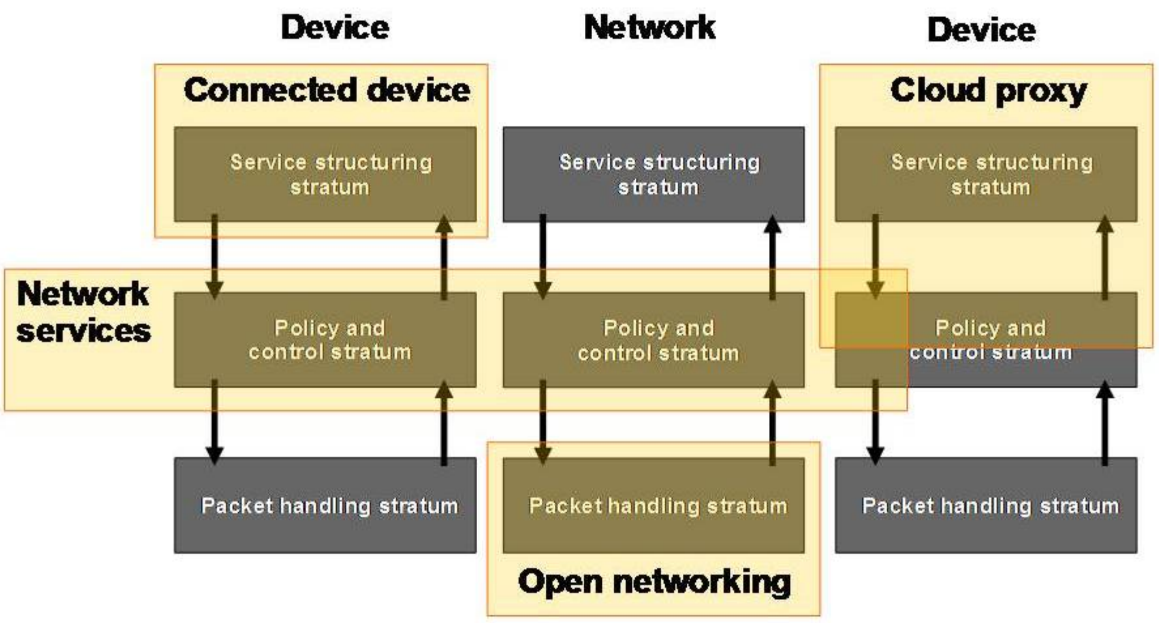
The architecture of communication networks can further be separated into three planes: data plane, control plane, and management plane. For illustration and for simplification the following description uses the "three Strata Model" without explicitly distinguishing their respective separation in the data plane, control, and management plane according to the NGN architecture.

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


Device
Network
Device
Connected device
Cloud proxy
Service structuring stratum
Service structuring stratum
Service structuring stratum
Network services
Policy and control stratum
Policy and control stratum
Policy and control stratum
Packet handling stratum
Packet handling stratum
Packet handling stratum
Open networking

Mapping of the Generic Enablers into the Communication Model

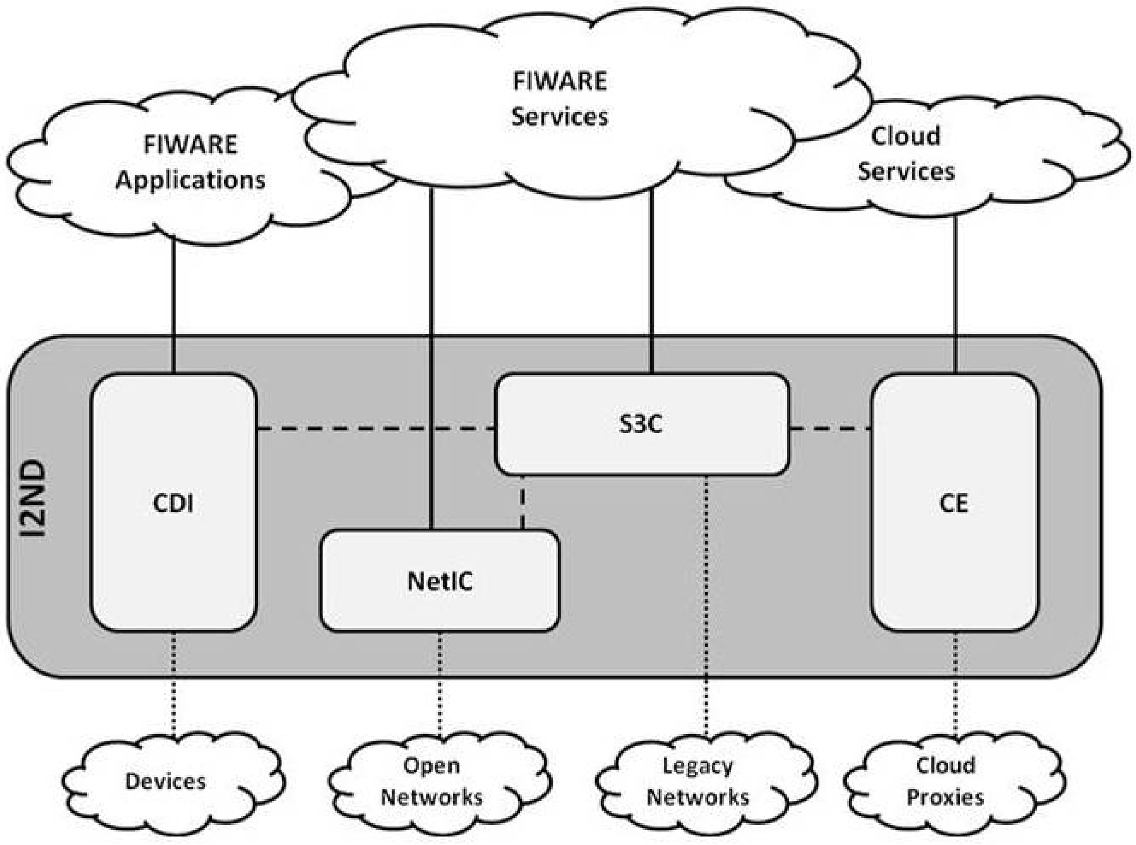
The I2ND chapter addresses the interfaces at the level of different strata. The Service Structuring Stratum includes interfaces towards applications in a wide range of Usage Areas. The chapter will not handle the implementation of the Service Structuring Stratum inside the network. The Packet Handling 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



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.

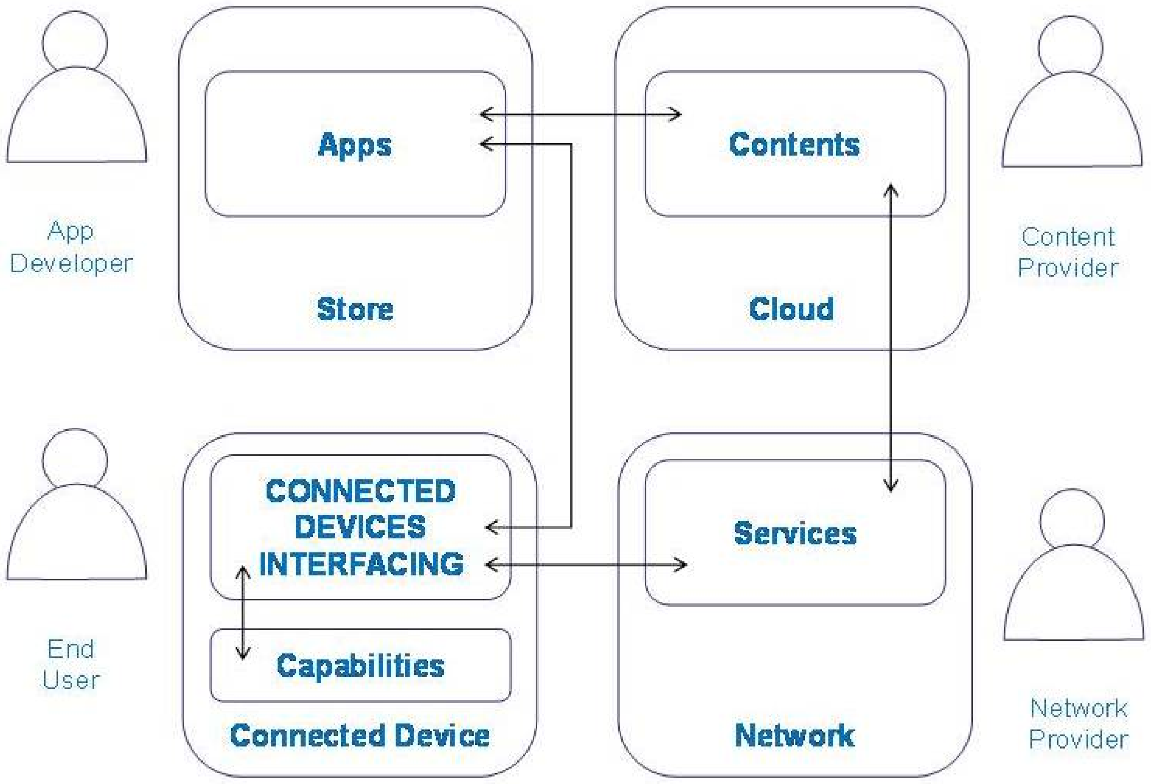
Generic Enablers

Connected Devices Interfacing (CDI)

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.



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

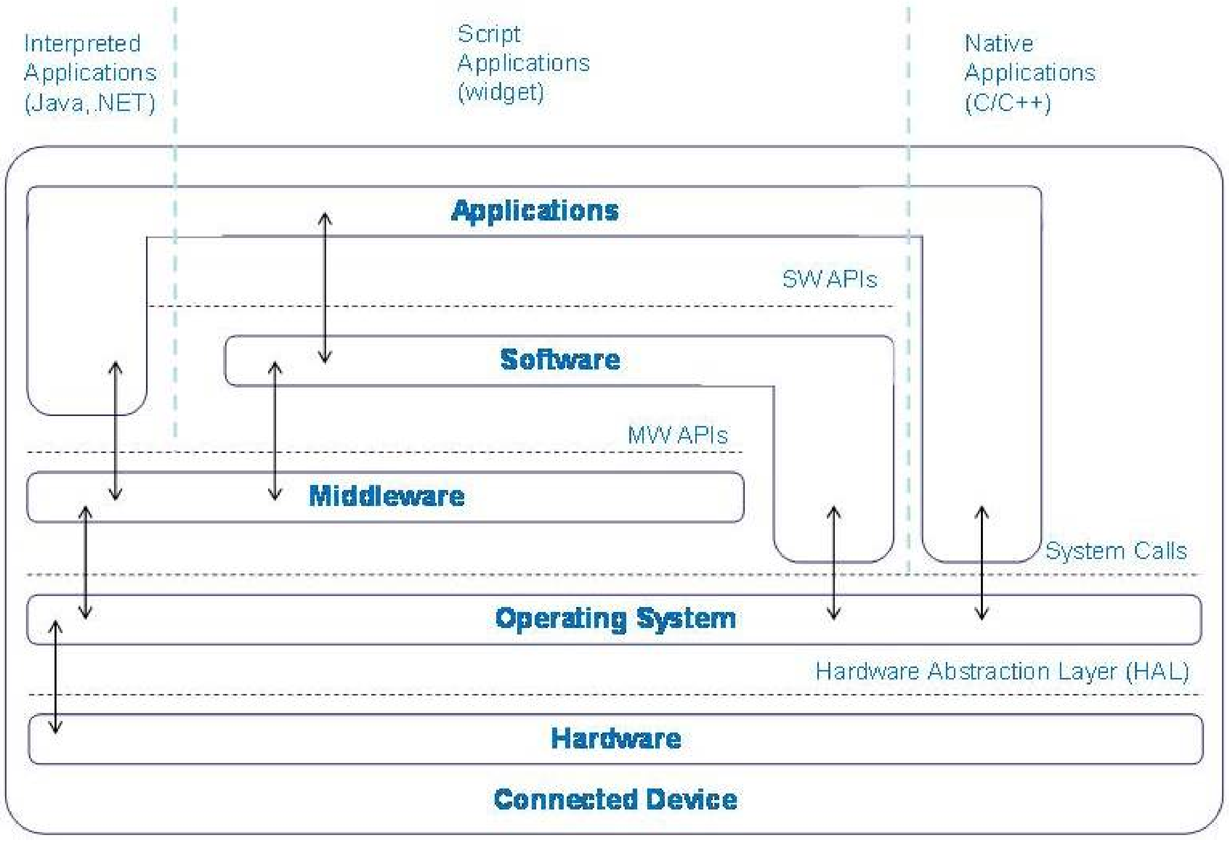
Another example might relate to location-based services, where the location of the device (shopping mall), might be a useful indication of the customer's intention, and suggest useful sidebar links in the form of advertising offers. Yet another might be the availability of triaxial accelerometers on 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.

GE description

Overview

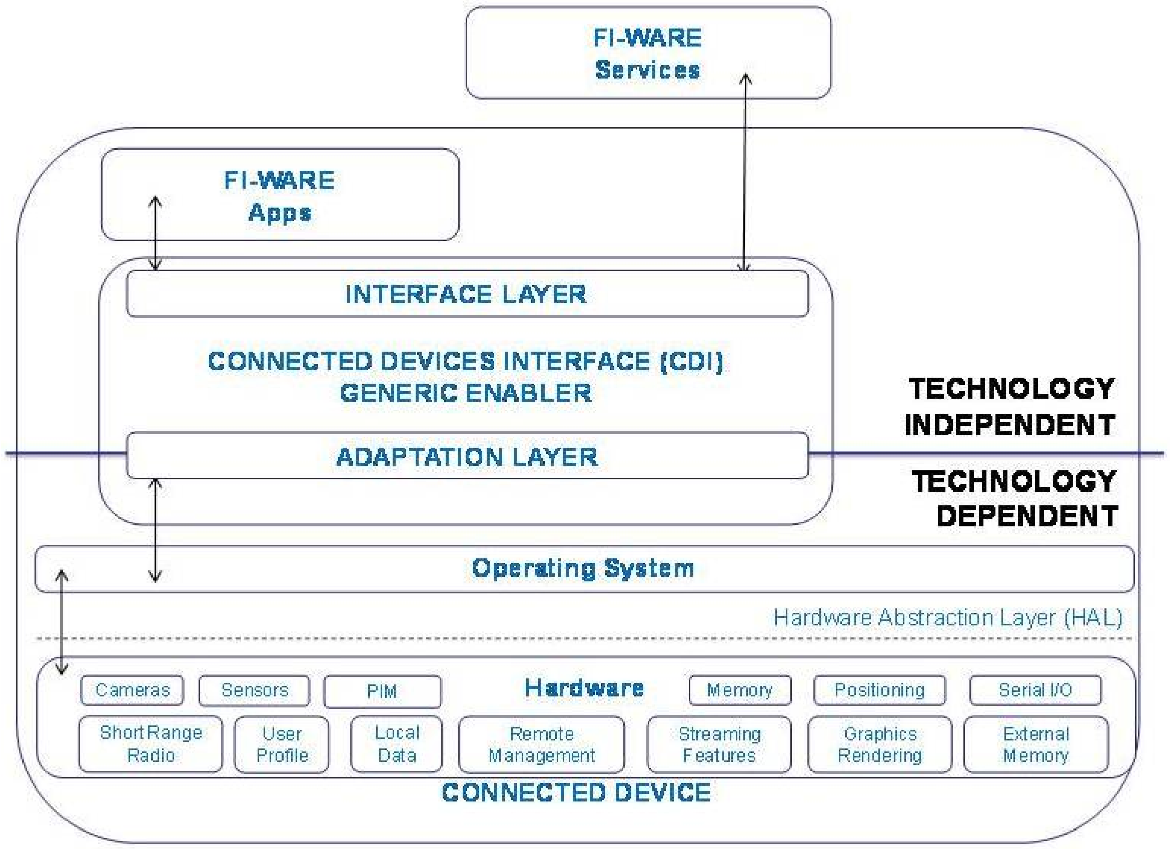
The CDI GE is in charge of addressing a broad set of connected devices, not only the mobile ones, each adopting specific technology solutions in terms of hardware, software, middleware and runtime platforms, in particular concerning the development of applications.

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.


Interpreted Applications (Java, .NET)
Script Applications (widget)
Native Applications (C/C++)
Applications
SW APIs
Software
MW APIs
Middleware
System Calls
Operating System
Hardware Abstraction Layer (HAL)
Hardware
Connected Device

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



CDI virtualisation features of the connected device

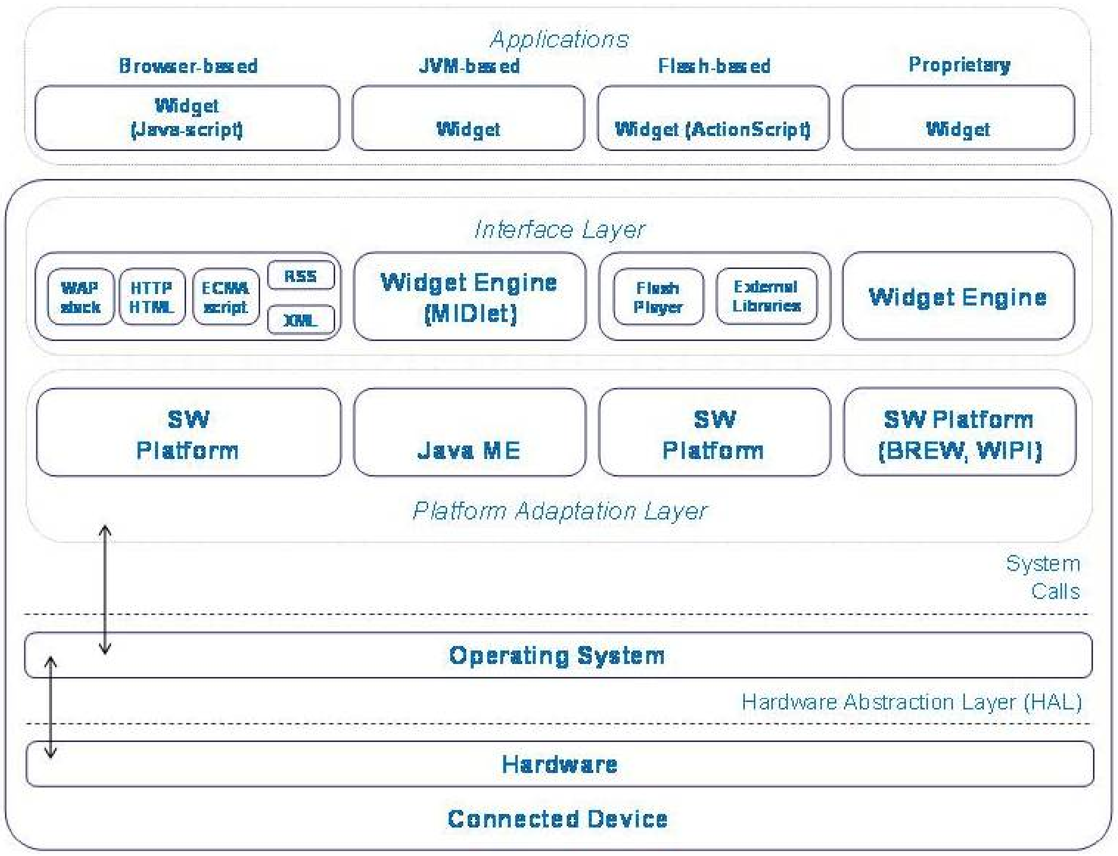
As shown in the figure, the CDI GE interfaces the FI-WARE services and applications with the connected device capabilities, such as: embedded sensors (cameras, GPS, accelerometer, compass, etc.), connectivity (short range radio, radio interfaces, wifi, bluetooth, streaming, etc.), data (PIM, user profile, 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.

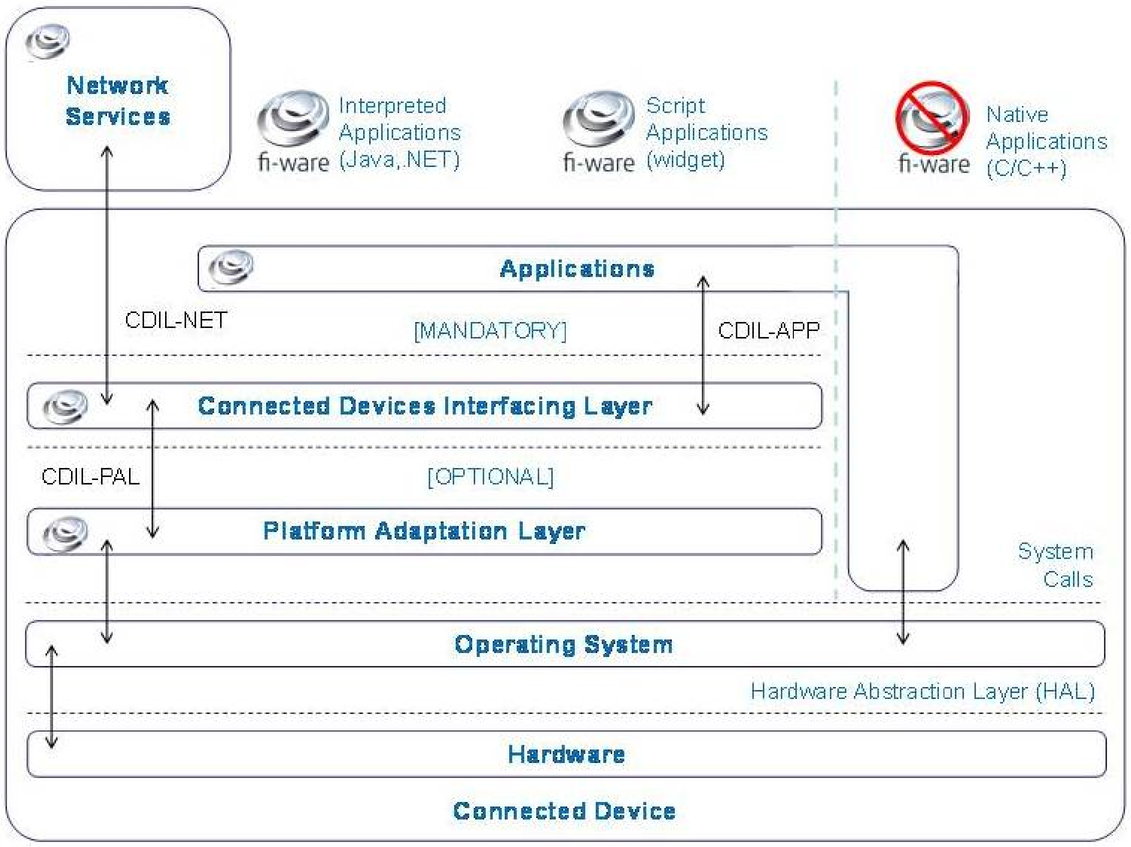
High level architecture

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:



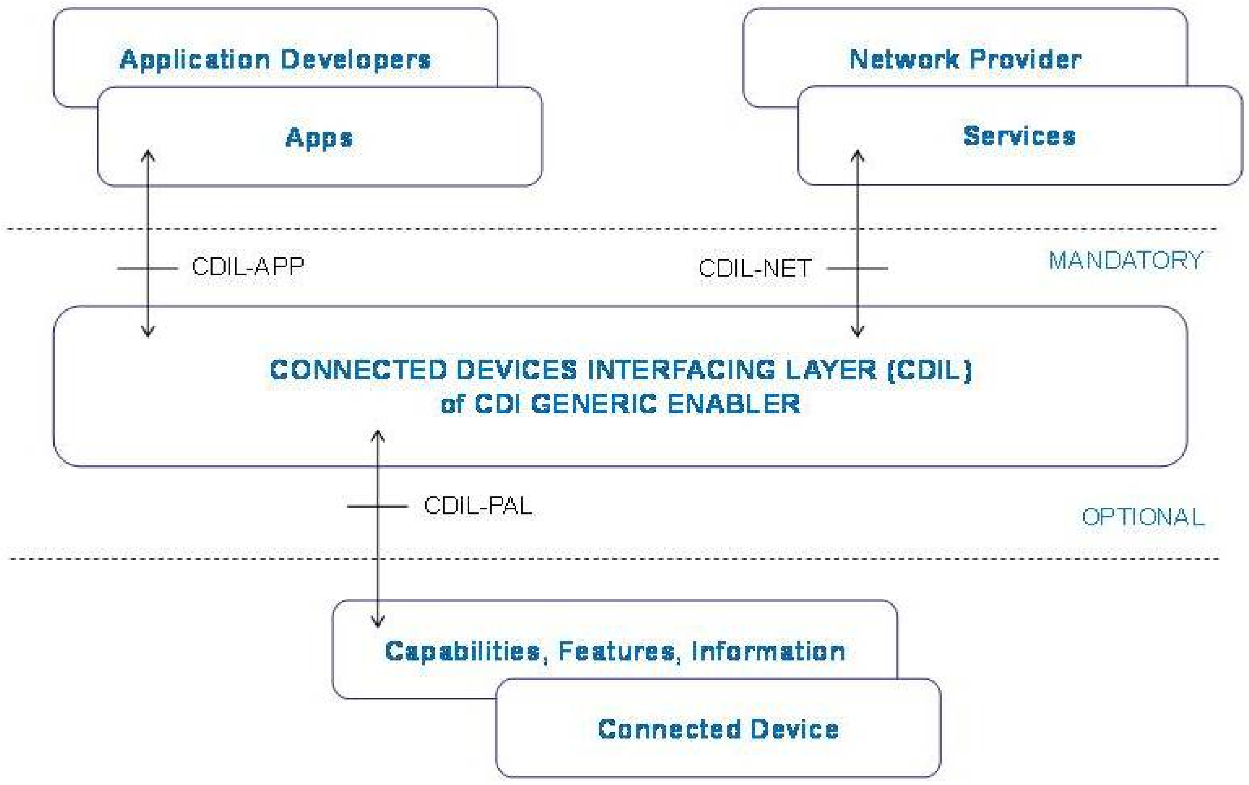
Reference execution environment for applications (widgets)

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.


Network Services
Interpreted Applications (Java,.NET)
fi-ware
Script Applications (widget)
fi-ware
Native Applications (C/C++)
fi-ware
Applications
CDIL-NET
[MANDATORY]
CDIL-APP
Connected Devices Interfacing Layer
CDIL-PAL
[OPTIONAL]
Platform Adaptation Layer
System Calls
Operating System
Hardware Abstraction Layer (HAL)
Hardware
Connected Device

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.



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.

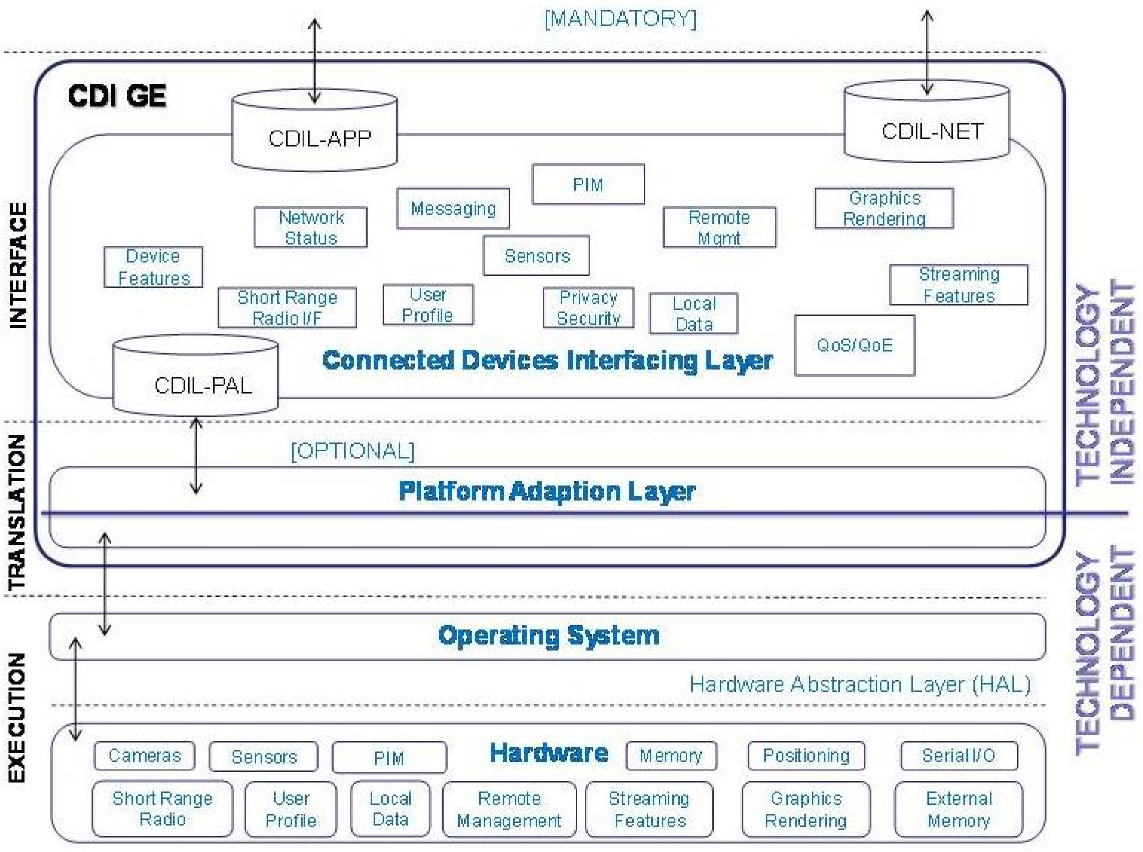
The CDIL can optionally rely on proper CDIL-PAL interfaces in order to interact with the connected device capabilities by means of an intermediate **Platform Adaptation Layer** (PAL) that virtualizes the heterogeneous device hardware and software capabilities into homogeneous, syntactically and semantically 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.



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, physical device properties, 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, physical device properties (touch UI, T9/Qwerty keyboard, stylus support, mouse, rollball, jog wheel), 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) and contactless 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, NFC).
  - **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** 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. The following set of graphical properties will be detected by the CDI:
    - 3D Hardware support (WebGL, OpenGL ES etc)
    - 3D Visual Display (e.g. LG OPtimus 3D P920 [1])
    - Physical Graphical Properties (Screen resolution, colour depth and DPI)
    - HD Out support (e.g. Nokia N8 [2])
    - Screen rotation support
  - **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. The two key run time device capabilties considered here are:
    - Video Codec Support (e.g. Quick Time, MP4, DivX, XCiVD, WMV, H.264, H263 etc)
    - Connection type and connection speed (Edge, 3G, HSDPA, LTE, WiFi, WiMAX etc)

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

Cloud Edge

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.

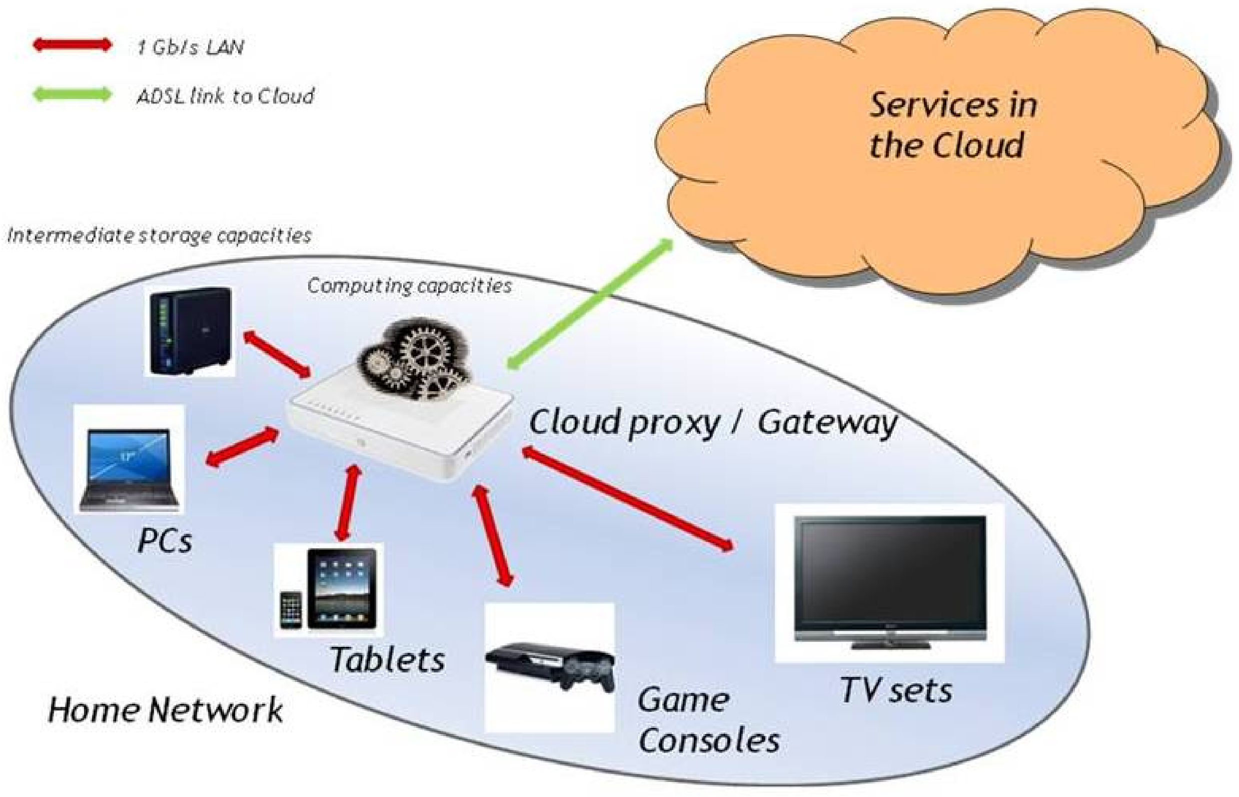
The concept of cloud proxy comes from the assessment that, despite the apparent inexhaustible computing or storage resources offered by the cloud concept, the link between the end-user and the cloud remains unique, providing a relative low bandwidth (compared to the internal home network bandwidth, in 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.

Lastly, the possible support in the cloud proxy of gaming acceleration features is also considered.

The following figure illustrates the concept of cloud proxy.



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.

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.

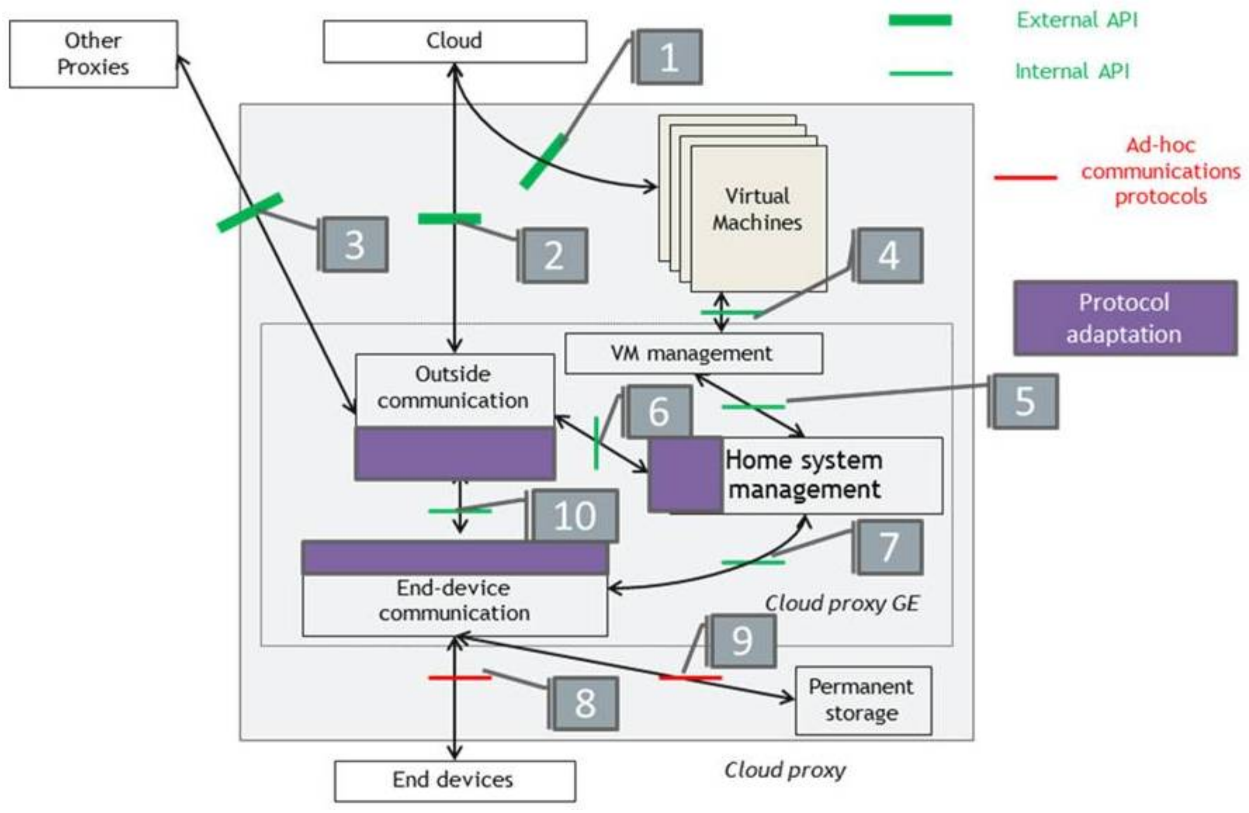
A set of APIs shall be defined, comprising API to the home system organisation module, basically for monitoring or resource description, and API to end-devices and permanent storage, basically for data exchange. The APIs may include the Connected Device Interfacing Layer (CDIL) as defined for the CDI 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.



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.

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.

Network Information and Control (NetIC)

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).

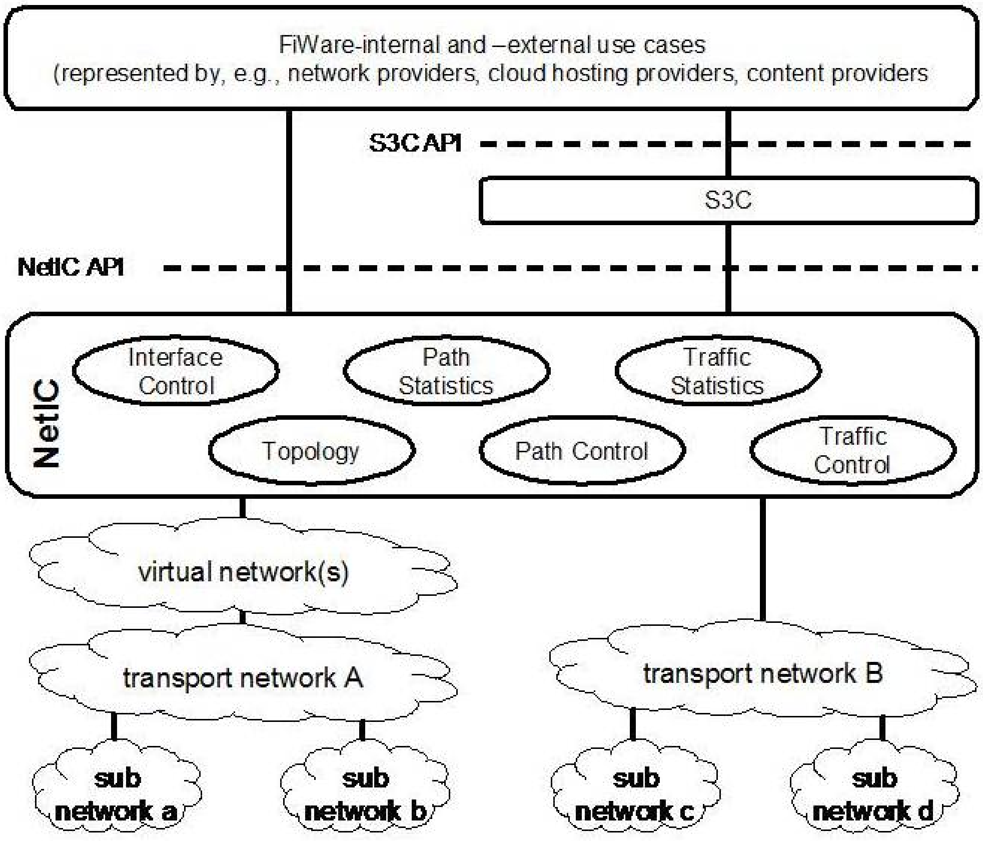
GE description

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.



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.

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.

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.

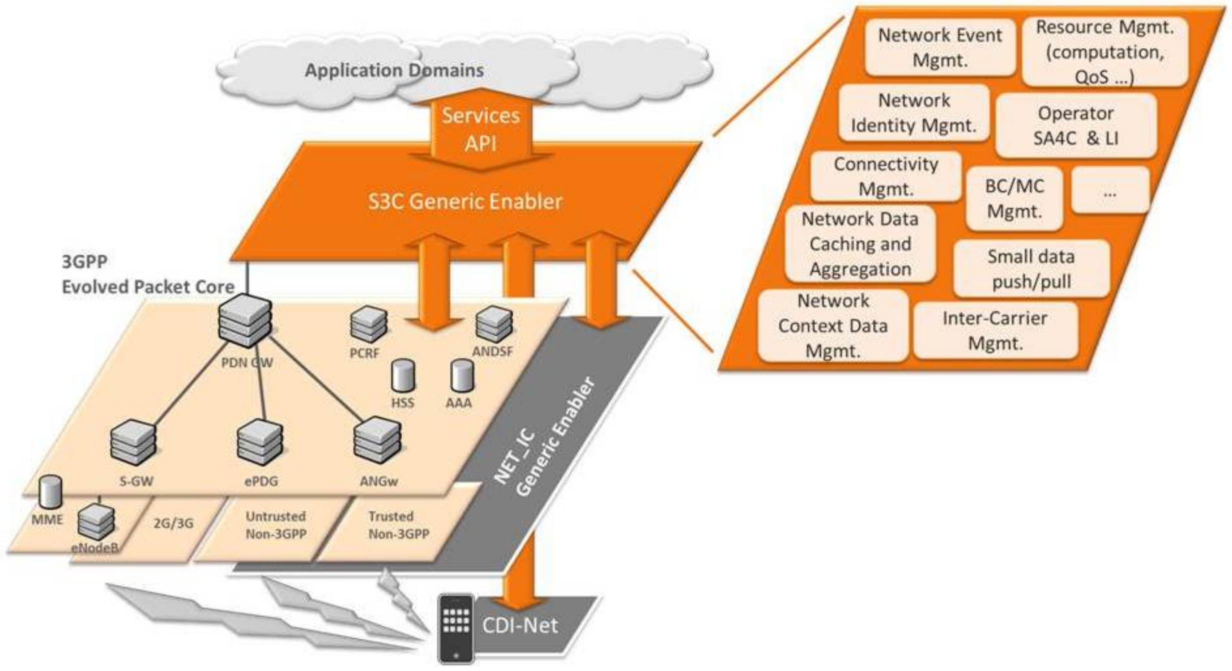
However, the EPC was designed with point-to-point human communication as the main service paradigm which will require a new adjustment in customizing the IP connectivity according to the momentary needs of a high number of devices using the broad future internet services.

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.



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.

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.

Question Marks

This section lists a number of questions that remain open. Further discussion on the questions will take place in the coming months.

Security aspects

Identity Management

Management of Identity of user is a 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 is however necessary a detailed analysis of the best solutions to be applied through the GE functionalities. Connected devices can offer additional functionality that can be used to improve the capabilities of the Security mechanisms. In particular, at least a subset of the devices (typically the mobile handsets, but this can be also the case of In-vehicle Infotainment units, or even some M2M modules) can provide specific capabilities offered by a secure element, which might be exploited to re-inforce the security functionality, e.g support the identity management. This can be obtained for instance through downloading with secure and standardised mechanisms Certificates, or other security elements (crypto keys, etc), as well as providing certified user profile information, which is made available in a controlled mode to the applications running on the Connected Device.

Secure Exposure of Control Functionality towards Application Platforms

The applications are able to control the data path offered by the network for the specific services synchronous to the communication provided that the application platforms have a business agreement with the core network provider. For example, this can be realized through the transmission of the communication requirements at the service establishment from the service platform to the core network. This issue does not relate to the actual data delivery between devices and application platforms. The exposure API of the core network which receives these requirements has to be able to identify and authorize the correct application platforms as well as to ensure that the communication over this exposure interface is established with the appropriate level of privacy. Also this interface should include the accounting and the validation of the Service Layer Agreements fulfilment from the side of the core network.

Secure Control Environment for Open Networking and Network Services

It is expected that the control environment which automatically executes decisions for the various functions part of the interface towards the networks is automatically validated through operation monitoring and behaviour evaluating the possible cyber-attacks in this part of the functionality. This includes the monitoring of data path traffic for open networking and network services, the abnormality detection and reporting towards the control functions for appropriate remediation. This is especially beneficial in case of self-organizing features which have to maintain the appropriate adaptation level according to the network circumstances.

Privacy Protection of Subscriber Information towards Applications

The Network Services GE exposes subscriber related information towards the FI-WARE internal and external use cases such as notifications on data path events and network context information such as device status, presence information, information on the vicinity of the device etc. This information exposure in most of the I2ND GEs (not only in Network Services but also in Connected Devices and Cloud Proxy) has to be controlled by means of rules, anonymized wherever necessary, and secured based on the requirements of the various applications and on the exposure levels allowed by the FI-WARE network operator. Also bulk data reporting should be possible in which all the information related to specific subscribers is converged under the unique identity of the network provider.

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.

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).

References

[IPsphere]	IPsphere project within the Telemanagement Forum, [3] [3]
[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.

References

[1] http://www.gsmarena.com/lg_optimus_3d_p920-3759.php
[2] http://www.nokia.com/ie-en/products/phone/n8-00/
[3] http://www.tmforum.org/ipsphere

Materializing Cloud Hosting in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Cloud Hosting chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how Cloud Hosting functions are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

Please also note that there are areas within Cloud Hosting that are common to other chapters within FIware that have yet to be agreed upon. These can be seen in this section.

IaaS Data Center Resource Management

Baseline Assets

- Crowbar and xCAT are physical infrastructure management and automation tools.
- OpenStack Nova is a resource management framework targeted at the management of virtualised compute-type resources.
- System Pools is an IBM technology allowing to intelligently manage pools of virtualized resources
- ISAAC is an IBM technology which provides resilient and scalable management fabric enablement in an OpenStack-like cloud stack
- RESERVOIR is an FP7 project completed in 2010, where we developed a series of cloud management technologies applicable to this GE
- Trusted Compute Pools provides additional functionality to the OpenStack Nova project that takes advantage of hardware based security features.
- Open Cloud Computing Interface (OCCI) is a RESTful protocol and API for the management of cloud service resources.
- OpenStack Glance is an image registry service that compliments OpenStack Nova.
- OpenStack Quantum is a service that provides advanced networking capabilities and services.
- Open vSwitch is a L2/L3 managed distributed virtual switch.

Epics

- FIWARE.Epic.Cloud.ResourceManager.Compute
- FIWARE.Epic.Cloud.ResourceManager.Compute.ImageRegistry
- FIWARE.Epic.Cloud.ResourceManager.Storage
- FIWARE.Epic.Cloud.ResourceManager.Network
- FIWARE.Epic.Cloud.ResourceManager.Metrics
- FIWARE.Epic.Cloud.ResourceManager.MgmtFabric
- FIWARE.Epic.Cloud.ResourceManager.QoS
- FIWARE.Epic.Cloud.ResourceManager.Mobility
- FIWARE.Epic.Cloud.ResourceManager.Placement
- FIWARE.Epic.Cloud.ResourceManager.PhysPlatformMgmt
- FIWARE.Epic.Cloud.ResourceManager.Capacity
- FIWARE.Epic.Cloud.ResourceManager.APIs
- FIWARE.Epic.Cloud.ResourceManager.SvcMgrIntegration
- FIWARE.Epic.Cloud.ResourceManager.Federation
- FIWARE.Epic.Cloud.ResourceManager.Security.APIs
- FIWARE.Epic.Cloud.ResourceManager.Security.SSO
- FIWARE.Epic.Cloud.ResourceManager.Security.RBAC
- FIWARE.Epic.Cloud.ResourceManager.Security.Compute
- FIWARE.Epic.Cloud.ResourceManager.Security.Network
- FIWARE.Epic.Cloud.ResourceManager.Security.Storage
- FIWARE.Epic.Cloud.ResourceManager.Setup.IBM
- FIWARE.Epic.Cloud.ResourceManager.Setup.Intel

Features

- FIWARE.Feature.Cloud.ResourceManager.Setup.IBM.BasicProvisioning
- FIWARE.Feature.Cloud.ResourceManager.Setup.Intel.BasicProvisioning
- FIWARE.Feature.Cloud.ResourceManager.APIs.EssentialManagement
- FIWARE.Feature.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone

User-Stories

- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.CreateVM
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.GetVMDetails
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.UpdateVM
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.ActionVM
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.DeleteVM
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.AddVMImage
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.GetVMImageDetails
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.DeleteVMImage
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.CreateVolume
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.AttachVolume
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.GetVolumeDetails
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.DetachVolume
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.DeleteVolume
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.CreateNetwork
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.GetNetworkDetails

- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.UpdateNetwork
- FIWARE.Story.Cloud.ResourceManager.APIs.EssentialManagment.DeleteNetwork
- FIWARE.Story.Cloud.ResourceManager.Setup.IBM.BasicProvisioning.Nova
- FIWARE.Story.Cloud.ResourceManager.Setup.IBM.BasicProvisioning.Glance
- FIWARE.Story.Cloud.ResourceManager.Setup.IBM.BasicProvisioning.Keystone
- FIWARE.Story.Cloud.ResourceManager.Setup.IBM.BasicProvisioning.Dashboard
- FIWARE.Story.Cloud.ResourceManager.Setup.Intel.BasicProvisioning.Nova
- FIWARE.Story.Cloud.ResourceManager.Setup.Intel.BasicProvisioning.Glance
- FIWARE.Story.Cloud.ResourceManager.Setup.Intel.BasicProvisioning.Keystone
- FIWARE.Story.Cloud.ResourceManager.Setup.Intel.BasicProvisioning.Dashboard
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.WorkloadExperiments
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.GUI
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.HistoryManager
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.EstimateCapacity
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.ApplyOverCommit
- FIWARE.Story.Cloud.ResourceManager.Capacity.SimpleDaytraderOvercommitStandalone.ValidateCompliance

IaaS Service Management

Baseline Assets

- The JBoss Enterprise Application Platform is the market leading platform for innovative and scalable Java applications.
- The MySQL™ software delivers a very fast, multi-threaded, multi-user, and robust SQL database server.
- The Claudia Platform is is an advanced Service Management toolkit that allows Service Providers to dynamically control the Service provisioning and scalability in an IaaS Cloud, supporting also the provisioning of PaaS and SaaS.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.Epic.Cloud.ServiceManager.Federation
- FIWARE.Epic.Cloud.ServiceManager.AggregateAPI
- FIWARE.Epic.Cloud.ServiceManager.ClaudiaScale
- FIWARE.Epic.Cloud.ServiceManager.Security

Features

- FIWARE.Feature.Cloud.ServiceManager.CDMI
- FIWARE.Feature.Cloud.ServiceManager.VDCManagement
- FIWARE.Feature.Cloud.ServiceManager.ServiceAutoScalling
- FIWARE.Feature.Cloud.ServiceManager.Snapshot
- FIWARE.Feature.Cloud.ServiceManager.VMClone
- FIWARE.Feature.Cloud.ServiceManager.Monitoring
- FIWARE.Feature.Cloud.ServiceManager.Template

User-Stories

- FIWARE.Story.Cloud.ServiceManager.CDMI.OVFextension
- FIWARE.Story.Cloud.ServiceManager.CDMI.CIMIextension
- FIWARE.Story.Cloud.ServiceManager.CDMI.CDMIClient
- FIWARE.Story.Cloud.ServiceManager.CDMI.ContainersMgmt

- FIWARE.Story.Cloud.ServiceManager.VDCManagement.VDCDeploy
- FIWARE.Story.Cloud.ServiceManager.VDCManagement.VDCUndeploy
- FIWARE.Story.Cloud.ServiceManager.VDCManagement.UpdateVDCConf

- FIWARE.Story.Cloud.ServiceManager.ServiceAutoScalling.GetMonitoringMetrics
- FIWARE.Story.Cloud.ServiceManager.ServiceAutoScalling.ElastRules
- FIWARE.Story.Cloud.ServiceManager.ServiceAutoScalling.AutomaticProcessingRules

- FIWARE.Story.Cloud.ServiceManager.AggregateAPI.OCCIClient
- FIWARE.Story.Cloud.ServiceManager.AggregateAPI.OVFCompatibility
- FIWARE.Story.Cloud.ServiceManager.AggregateAPI.CIMIImplementation

- FIWARE.Story.Cloud.ServiceManager.Snapshot.CreateSnapshot
- FIWARE.Story.Cloud.ServiceManager.Snapshot.RestoreSnapshot
- FIWARE.Story.Cloud.ServiceManager.Snapshot.DeleteSnapshot

- FIWARE.Story.Cloud.ServiceManager.VMClone.CreateVMClone
- FIWARE.Story.Cloud.ServiceManager.VMClone.RestoreVMClone
- FIWARE.Story.Cloud.ServiceManager.VMClone.DeleteVMClone

- FIWARE.Story.Cloud.ServiceManager.Monitoring.VMProbeInstalling
- FIWARE.Story.Cloud.ServiceManager.Monitoring.ColletcData
- FIWARE.Story.Cloud.ServiceManager.Monitoring.PublishSubscribe

- FIWARE.Story.Cloud.ServiceManager.Security.SSO
- FIWARE.Story.Cloud.ServiceManager.Security.RBAC

- FIWARE.Story.Cloud.ServiceManager.Template.GetTemplateInformation
- FIWARE.Story.Cloud.ServiceManager.Template.VM2Template

PaaS Management

Baseline Assets

The PaaS Generic Enabler will be based on the outcomings of the 4CaaSt (FP7-257928) project [1], a European Commission FP7 funded Integrated Project started on June 1st 2010. Some of the baseline assets that we can give you in advance are the following:

- The Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.
- The PostgreSQL is open source object-relational database system.
- The Ruby is open source programming language with a focus on simplicity and productivity.
- The RubyGem software itself allows you to easily download, install, and manipulate software package on your system.
- The Chef is an open-source systems integration framework built specifically for automating the cloud.
- PaaS Management Platform is is an advanced PaaS Management toolkit that allows 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.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.Epic.Cloud.PaaS.OVF+
- FIWARE.Epic.Cloud.PaaS.PaaSScaling
- FIWARE.Epic.Cloud.PaaS.ApplicationMgmt
- FIWARE.Epic.Cloud.PaaS.PaaSConfigurationMgmt
- FIWARE.Epic.Cloud.PaaS.InstalationMgmt
- FIWARE.Epic.Cloud.PaaS.PlatformMgmt
- FIWARE.Epic.Cloud.PaaS.DeploymentDesign
- FIWARE.Epic.Cloud.PaaS.Monitoring
- FIWARE.Epic.Cloud.PaaS.Security

Object Storage

Baseline Assets

- OpenStack Swift is a highly available, distributed, eventually consistent object/blob store.

Epics

- FIWARE.Epic.Cloud.ObjectStorage.APIs
- FIWARE.Epic.Cloud.ObjectStorage.Security.SSO
- FIWARE.Epic.Cloud.ObjectStorage.Security.RBAC
- FIWARE.Epic.Cloud.ObjectStorage.Security.IrrevocableErasure
- FIWARE.Epic.Cloud.ObjectStorage.Security.EncryptedStorage
- FIWARE.Epic.Cloud.ObjectStorage.Monitoring
- FIWARE.Epic.Cloud.ObjectStorage.Versioning

Features

- FIWARE.Feature.Cloud.ObjectStorage.APIs.EssentialManagment
- FIWARE.Feature.Cloud.ObjectStorage.Setup.Intel.BasicObjectStorage

User-Stories

- FIWARE.Story.Cloud.ObjectStorage.Setup.Intel.BasicObjectStorage.Swift
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.AddObject
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.GetObject
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.GetObjectDetails
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.UpdateObject
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.UpdateObjectDetails
- FIWARE.Story.Cloud.ObjectStorage.APIs.EssentialManagment.DeleteObject

Self-Service Interfaces

Baseline Assets

- Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.
- MySQL™ software delivers a very fast, multi-threaded, multi-user, and robust SQL database server.
- HTML is the predominant markup language for web pages. HTML elements are the basic building-blocks of webpages.
- CSS is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML.
- Ruby on Rails is an open source web application framework for the Ruby programming language

Epics

- FIWARE.Epic.Cloud.SelfServiceInterfaces.UserInterface
- FIWARE.Epic.Cloud.SelfServiceInterfaces.AdminInterface
- FIWARE.Epic.Cloud.SelfServiceInterfaces.Security

Features

- FIWARE.Feature.Cloud.SelfServiceInterfaces.UserInterface.UserPortal
- FIWARE.Feature.Cloud.SelfServiceInterfaces.UserInterface.UserCLI
- FIWARE.Feature.Cloud.SelfServiceInterfaces.AdminInterface.AdminCLI
- FIWARE.Feature.Cloud.SelfServiceInterfaces.Security.Authentication
- FIWARE.Feature.Cloud.SelfServiceInterfaces.Security.Authorization

User-Stories

- FIWARE.Story.Cloud.SelfServiceInterfaces.UserInterface.UserPortal.UseCaseSpecification
- FIWARE.Story.Cloud.SelfServiceInterfaces.UserInterface.UserPortal.ServiceManagerAPIAnalysis
- FIWARE.Story.Cloud.SelfServiceInterfaces.UserInterface.UserPortal.UseCaseVerification
- FIWARE.Story.Cloud.SelfServiceInterfaces.UserInterface.UserPortal.GUIDesign
- FIWARE.Story.Cloud.SelfServiceInterfaces.Security.Authentication.SSO
- FIWARE.Story.Cloud.SelfServiceInterfaces.Security.Authorization.RBAC

Cloud Proxy

Baseline Assets

- The Nada Management Framework is a set of client/server modules used to remotely configure an Application to run on virtual host running on a set of virtualized gateways.

Epics

- FIWARE.Epic.Cloud.CloudProxy.ServicePlatformManager
- FIWARE.Epic.Cloud.CloudProxy.VirtualEnvironmentSystem
- FIWARE.Epic.Cloud.CloudProxy.Monitoring

Features

- FIWARE.Feature.Cloud.CloudProxy.APIFirstRelease

Monitoring

Baseline Assets

- collectd is a daemon which collects system performance statistics periodically and provides mechanisms to store the values in a variety of ways, for example in RRD files.
- RabbitMQ provides robust messaging for applications. It is easy to use, fit for purpose at cloud scale and supported on all major operating systems and developer platforms.
- Drools is a Business Logic integration platform.
- Esper is a complex processing framework and runtime.
- Graphite is a monitoring metric storage and visualisation framework and runtime.

Epics

- FIWARE.Epic.Cloud.Monitoring.SourceMetrics
- FIWARE.Epic.Cloud.Monitoring.Distribution
- FIWARE.Epic.Cloud.Monitoring.Storage
- FIWARE.Epic.Cloud.Monitoring.Analysis
- FIWARE.Epic.Cloud.Monitoring.Analysis.SLAIntegration
- FIWARE.Epic.Cloud.Monitoring.Visualization
- FIWARE.Epic.Cloud.Monitoring.APIs
- FIWARE.Epic.Cloud.Monitoring.Security.SSO
- FIWARE.Epic.Cloud.Monitoring.Security.RBAC
- FIWARE.Epic.Cloud.Monitoring.Security.Metrics

Features

- FIWARE.Feature.Cloud.Monitoring.ResourceManager.Compute
- FIWARE.Feature.Cloud.Monitoring.ResourceManager.Storage
- FIWARE.Feature.Cloud.Monitoring.ResourceManager.Network
- FIWARE.Feature.Cloud.Monitoring.ObjectStorage

Security

Baseline Assets

- OpenStack Keystone is an identity service providing SSO and RBAC.

Themes

- Security Matrix [2]

Epics

- FIWARE.Epic.Cloud.Security.APIs
- FIWARE.Epic.Cloud.Security.SSO
- FIWARE.Epic.Cloud.Security.RBAC

Topics still being addressed at high-level

- Security
- Monitoring
- Accounting

References

[1] http://4caast.morfeo-project.org/
[2] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php?title=CHSecurity

Crowbar

Brief Description

Crowbar is a physical infrastructure management tool that provides the means to automate the deployment of infrastructural services to physical hosts. Crowbar consists mainly of a DHCP, DNS, PXE boot server and service based on OpsCode Chef [1]. Once Crowbar is running and a new physical node boots via the network (PXE boot), a custom Centos image is loaded and then enters into an advertisment mode. Crowbar discovers the newly advertised and then upon the command of the administrator, can apply various roles to a physical server (e.g. an OpenStack Nova compute node or an OpenStack swift storage node). These roles are known as barclamps and new barclamps can be implemented with a combination of Ruby and the Chef framework [1].

IPR

Crowbar is licensed [2] under the Apache 2.0 license [3].

Publically Available Documentation

- The architect of Crowbar has posts detailing aspects of Crowbar [4]
- Crowbar Wiki [5]
- Creating a barclamp [6]
- Crowbar Videos [7]
- Crowbar Source [8]

References

[1] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Chef
[2] http://robhirschfeld.com/2011/07/26/crowbar-source-released/
[3] http://www.apache.org/licenses/LICENSE-2.0.html
[4] http://robhirschfeld.com/tag/crowbar/
[5] https://github.com/dellcloudedge/crowbar/wiki
[6] https://github.com/dellcloudedge/crowbar/wiki/Barclamp%3A-create-%26-install-steps
[7] http://www.youtube.com/user/rollingwoodcitizen
[8] https://github.com/dellcloudedge/crowbar

XCAT

Brief Description

xCAT is a physical infrastructure management and provisioning tool that provides the means to automate high-scale deployment of infrastructural services to physical hosts, as well as to perform hardware management and monitoring.

IPR

xCAT is licensed under Eclipse Public License [1]

Publically Available Documentation

- xCAT home page [2]

References

[1] http://www.eclipse.org/legal/epl-v10.html
[2] http://xcat.sourceforge.net/

OpenStack Nova

Brief Description

OpenStack Nova is a resource management framework targeted at the management of virtualised compute-type resources. OpenStack aims to be independent from hardware and hypervisor choices of a particular service provider. As well as managing, both through its API and Portal, the compute resources it also has the capabilities to manage networking and storage resources associated with the compute resources. A good overview of the various features of Nova [1] can be found on the OpenStack website.

Nova can be integrated into the Single Sign On service, Keystone.

Hypervisors Supported:

- Hyper-V 2008 [2]
- VMWare ESX/ESXi [3]
- Xen [4]
- KVM [5]
- QEMU [6]
- LXC [7]
- UML [8]

Networking Schemes Supported:

- Flat [9]
- Flat DHCP [9]
- VLAN Network [9]

OpenStack Nova also supports the automated creation of L2/L3 network bridges that can be used to implement cloud-bursting scenarios and also secure access to a tenants compute resources. This is implemented with the Cloudpipe feature [10] of OpenStack.

Storage and Volume Management Support:

- iSCSI [11]

OpenStack can uses the open source Open-iSCSI implementation [12]

IPR

OpenStack Nova is licensed [13] under the Apache 2.0 license [3].

Publically Available Documentation

- Overview [1]
- OpenStack Nova Administration Guide [14]
- OpenStack Nova Developer and Architecture Documentation [15]
- OpenStack Nova API [16]
- OpenStack Nova Source [17]

References

[1] http://openstack.org/projects/compute/
[2] https://www.microsoft.com/windowsserver2008/en/us/hyperv-main.aspx
[3] http://www.vmware.com/products/vsphere-hypervisor/support.html
[4] http://www.xen.org/support/documentation.html
[5] http://www.linux-kvm.org/page/Main_Page
[6] http://wiki.qemu.org/Manual
[7] http://lxc.sourceforge.net/
[8] http://user-mode-linux.sourceforge.net/
[9] http://docs.openstack.org/diablo/openstack-compute/admin/content/networking-options.html
[10] http://docs.openstack.org/diablo/openstack-compute/admin/content/cloudpipe-per-project-vpns.html
[11] http://en.wikipedia.org/wiki/ISCSI
[12] http://www.open-iscsi.org/
[13] http://www.openstack.org/projects/openstack-faq/
[14] http://docs.openstack.org/diablo/openstack-compute/admin/content/
[15] http://nova.openstack.org/
[16] http://docs.openstack.org/api/openstack-compute/1.1/content/
[17] https://github.com/openstack/nova

System Pools

Brief Description

The System Pools technology allows to intelligently manage pools of virtualized resources -- including provisioning as well as runtime management and optimization. It is a key asset behind the Enterprise Edition of "IBM Systems Director VMControl" product [1].

Programming Artifacts

System Pools functionality will be used via REST APIs.

Technologies Used

The software is implemented in Java

Runtime Prerequisites

- Java 6

IPR

System Pools is an IBM proprietary technology. It will be licensed under FRAND [2] terms according to the FI-PPP program rules.

Publically Available Documentation

- System Pools in IBM Systems Director VMControl [3]

References

[1] http://www-03.ibm.com/systems/software/director/vmcontrol/enterprise.html
[2] http://en.wikipedia.org/wiki/Fair,_reasonable,_and_non-discriminatory_terms
[3] http://publib.boulder.ibm.com/infocenter/director/v6r2x/index.jsp?topic=%2Fcom.ibm.director.vim.helps.doc%2Ffsd0_vim_t_managing_pools.html

ISAAC

Brief Description

ISAAC is a management fabric that allows to introduce high scalability and availability to OpenStack-like cloud management stacks. It is the core asset behind the "IBM Service Agility Accelerator for Cloud" [1] product.

Programming Artifacts

ISAAC functionality will be used via REST APIs.

Technologies Used

The software is mostly implemented in Ruby.

Runtime Prerequisites

Ruby

IPR

ISAAC is an IBM proprietary technology. It will be licensed under FRAND [2] terms according to the FI-PPP program rules.

Publically Available Documentation

- IBM Service Agility Accelerator for Cloud [2]

References

[1] http://www-01.ibm.com/software/tivoli/products/smartcloud-provisioning/
[2] http://publib.boulder.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.hslt.doc_1.0.0/welcome.html

RESERVOIR

Brief Description

RESERVOIR is an FP7 project, which has developed the following technologies applicable to this GE:

- Enhanced VM mobility
- Federated placement
- Admission control

IPR

The above technologies were developed in IBM, and will be licensed under FRAND [2] terms according to the FI-PPP program rules.

Publically Available Documentation

- RESERVOIR homepage [1]

References

[1] http://62.149.240.97/

Trusted Compute Pools

Brief Description

The Trusted Compute Pool (TCP) provides additional functionality to the OpenStack Nova project that takes advantage of hardware based security features such as TPM [1] and TXT [2]. TCP provides the ability to measure software components (e.g. virtual machine components or images) and store those measurements in cryptographic hardware. Complimenting this is remote verification (attestation) carried out be a separate entity. This allows for 3rd party trust providers attest that software components have not been modified.

TCP is a contribution being made to OpenStack from Intel.

IPR

TCP is licensed [13] under the Apache 2.0 license [3].

Publically Available Documentation

- TCP Overview [3]
- TCP OpenStack Blueprint [4]
- OpenStack Glance Developer and Architecture Documentation [5]
- TCP Source [6]
- TCP Code Review [7]

References

[1] http://en.wikipedia.org/wiki/Trusted_Platform_Module
[2] http://en.wikipedia.org/wiki/Trusted_Execution_Technology
[3] http://wiki.openstack.org/TrustedComputingPools
[4] https://blueprints.launchpad.net/nova/+spec/trusted-computing-pools
[5] http://glance.openstack.org/
[6] https://code.launchpad.net/~fred-yang/nova/TrustedComputingPools
[7] https://review.openstack.org/#change,675

Open Cloud Computing Interface (OCCI)

Brief Description

The 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. Good coverage of OCCI can be found in the Cloud Hosting Product Vision [1]. A good walkthrough of OCCI [2] can be found on the OCCI website.

IPR

The OCCI specification is licensed [13] under the IPR provisions [3] of OGF [4], which is considered to be FRAND. Please see the individual licenses of each of the above listed open source projects.

Publically Available Documentation

- OCCI Core Model [5]
- OCCI Infrastructure Model [6]
- OCCI HTTP Rendering [7]
- OCCI Monitoring and SLA draft specification [8]
- OCCI Open Source Projects [9]
- occi-py: OCCI Python Implementation [10]
- Pyssf: OCCI Python Implementation [11]
- OCCI for OpenNebula (decoupled ruby implementation) [12]
- occi-grammar: OCCI grammar [13]

References

[1] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Cloud_Hosting#GE_description
[2] http://occi-wg.org/2011/06/08/occi-presentation-a-walkthrough-of-the-specification/
[3] http://www.ogf.org/merger_docs/OGFIPRPolicy.pdf
[4] http://www.ogf.org
[5] http://docs.ogf.org/documents/GFD.183.pdf
[6] http://docs.ogf.org/documents/GFD.184.pdf
[7] http://docs.ogf.org/documents/GFD.185.pdf
[8] https://docs.google.com/document/d/1nWqOHuwpDnMVDYzDx2VxZ2ym79aU1W4laqj6-_FuYxs/edit?authkey=CI-lg6gJ&hl=en_US&authkey=CI-lg6gJ
[9] https://github.com/occi
[10] https://github.com/occi/occi-py
[11] https://github.com/occi/pyssf
[12] http://opennebula.org/software:ecosystem:occi
[13] https://github.com/occi/occi-grammar

OpenStack Glance

Brief Description

OpenStack Glance is an image registry service that compliments OpenStack Nova. It has the features of discovery, registration and access for virtual machine images. Glance supports many different virtual machine images and those images can be either made public or private. A good overview of the various features of Glance [1] can be found on the OpenStack website.

Nova can be integrated into the Single Sign On service, Keystone.

Supported Virtual Machine Images:

- OVF (VMWare, others)
- Raw
- qcow2 (Qemu/KVM)
- Machine (kernel/ramdisk outside of image, a.k.a. AMI)
- VDI (VirtualBox)
- VMDK (VMWare)
- VHD (Hyper-V)

Supported Storage Backends:

- Filesystem [2]
- Swift [3]
- Amazon S3 [4]

IPR

OpenStack Glance is licensed [13] under the Apache 2.0 license [3].

Publically Available Documentation

- Overview [1]
- OpenStack Glance Administration Guide [5]
- OpenStack Glance Developer and Architecture Documentation [5]
- OpenStack Glance API [6]
- OpenStack Glance Source [7]

References

[1] http://www.openstack.org/projects/image-service/
[2] http://docs.openstack.org/diablo/openstack-image-service/admin/content/configuring-the-filesystem-storage-backend.html
[3] http://docs.openstack.org/diablo/openstack-image-service/admin/content/configuring-the-swift-storage-backend.html
[4] http://docs.openstack.org/diablo/openstack-image-service/admin/content/configuring-the-s3-storage-backend.html
[5] http://docs.openstack.org/diablo/openstack-image-service/admin/content/
[6] http://docs.openstack.org/api/openstack-image-service/1.0/content/
[7] https://github.com/openstack/glance

OpenStack Quantum

Brief Description

OpenStack Quantum is a service that provides advanced networking capabilities beyond what is provided by OpenStack Nova (Flat, Flat DHCP, VLAN Network). It allow for the creation of network topologies most often associated with multi-tier applications. It provides a means of networking services such as VPN, IDS etc.

Supported L2/L3 Networking Soft/Hardware

- Open vSwitch [1] (Overview)
- Cisco Nexus [2]
- Cisco UCS Blades [3]

IPR

OpenStack Quantum is licensed [13] under the Apache 2.0 license [3].

Publically Available Documentation

- Overview [4]
- OpenStack Quantum Administration Guide [5]
- OpenStack Quantum Design Documentation [6]
- OpenStack Quantum API [7]
- OpenStack Quantum Source [8]

References

[1] https://github.com/openstack/quantum/tree/master/quantum/plugins/openvswitch
[2] https://github.com/openstack/quantum/tree/master/quantum/plugins/cisco/nexus
[3] https://github.com/openstack/quantum/tree/master/quantum/plugins/cisco/ucs
[4] http://wiki.openstack.org/Quantum
[5] http://docs.openstack.org/incubation/openstack-network/admin/content/
[6] http://wiki.openstack.org/QuantumOverview
[7] http://docs.openstack.org/incubation/openstack-network/developer/quantum-api-1.0/content/
[8] https://github.com/openstack/quantum

Open vSwitch

Brief Description

Open vSwitch is a L2/L3 managed distributed virtual switch. It provides inter- and intra-physical host virtual networking capabilities. Can allow for cross-subnet migrations via virtualisation. Open vSwitch can be implemented as a completely software-only solution but also a hybrid one where performance critical components are run upon a physical network card (e.g. via VMDq [1] facilities). It exports Linux bridging & VDE compatible interfaces and operates with almost identical network performance [2]. It also has support for OpenFlow sflow and NetFlow protocols and was patched recently [3] to support VXLAN [4]. A good overview of the various features of Open vSwitch [5] can be found on the Open vSwitch website.

IPR

Open vSwitch is licensed [6] under the Apache 2.0 license [3].

Publically Available Documentation

- General [5]
- Open vSwitch Documentation [7]
- Open vSwitch Source [8]

References

[1] http://www.intel.com/content/www/us/en/network-adapters/gigabit-network-adapters/io-acceleration-technology-vmdq.html
[2] http://openvswitch.org/papers/hotnets2009.pdf
[3] http://networkheresy.wordpress.com/2011/10/13/the-first-open-vswitch-vxlan-patch-is-in/
[4] http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00
[5] http://openvswitch.org/features/
[6] http://openvswitch.org/development/
[7] http://openvswitch.org/support/
[8] http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch

Claudia

Brief Description

Claudia is a standard-based service manager to easily govern whole services (rather than individual VMs) and to control how services are scaled up/down and in/out in an automated manner. Claudia helps to save valuable time/resources by automating prone to failure tasks that need to be done repeatedly by APs. Operations are done once for the whole service, rather than as many times as the number of required VMs for a given service.

Claudia offers an IaaS solution that lets the service providers define their service architecture and images (using a standard DMTF OVF descriptor with some extensions) and then deploy and manage them as a whole entity, inside a virtualization provider, controlling the configuration of multi-vm components, virtual networks and storage support by optimizing the usage of them by dynamically scaling services applying elasticity rules, SLA and business rules. The service definition includes the hardware requirements needed for each service element as well as for shared service networks, in order to assign the hardware requirements needed for the full stack.

Claudia can deploy services among a public Cloud (Amazon, Flexiscale, GoGrid, …) or a private Cloud using Virtual Infrastructure Manager (such as OpenNebula, Eucalyptus, vSphere, …) through a plug-in driver mechanism that will orchestrate the virtual resource allocation.

Programming artifacts

The Claudia platform will implement a DMTF's Cloud Infrastructure Management Interface (CIMI) (currently Work in Progress), using inside the DMTF’s Open Virtualization Format (OVF) to the upper layer. Claudia Platform also implements OCCI interface using also inside OVF specification of virtual appliances. Both of them are bindings in REST/HTTP

Technologies used

The Claudia platform is developed using J2EE. Claudia is installed on JBoss through a generated .ear. Libraries needed to work with is provided by this distribution and the technologies used are:

- J2EE 5.0
  - EJB 3.0
  - JPA 2.0 + Hibernate 3.3.0
  - JAX-B
  - JAX-RS
  - JMS
  - MDB
- Drools Expert 5.2
- RIF (W3C Working Group Note 22 June 2010)

Runtime pre-requisites

The Claudia platform has been built and tested on Windows 7, Debian Squeeze (Debian GNU/Linux 6.0, kernel 2.6.32-30) and Mac OS X 10.6.8.

Known software requirements

- JBoss 5.1 EAP
- MySQL 5.1.57
- JDK >= 1.5

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) [2] Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

- Drools Expert User Guide [1]
- TCloud API Specification [2]

(disclaimer) Although TCloud API is the current available interface we propose to change it to a standard solution specified by DMTF's Cloud Infrastructure Management Interface (CIMI) Primer (work in progress) [3]

References

[1] http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/index.html
[2] http://www.tid.es/files/doc/apis/TCloud_API_Spec_v0.9.pdf
[3] http://www.google.es/url?sa=t&rct=j&q=dmtf%20cimi&source=web&cd=1&ved=0CCEQFjAA&url=http%3A%2F%2Fwww.dmtf.org%2Fsites%2Fdefault%2Ffiles%2Fstandards%2Fdocuments%2FDSP2027_1.0.0a.pdf&ei=om6hTvfHH8K08QOB3qnPBQ&usg=AFQjCNGp4McT4VSZiruJz-4ahjI4C2yH3Q&cad=rja

PaaS Management Platform

Brief Description

The PaaS Management Platform provides 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 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.

The PaaS Management Platform is divided into two components, PaaS Governance with controls are the intelligence of the PaaS like the scaling rules and “recipes”. This recipes describe the packages that should be installed and services that should be running for example and will be sent to the PaaS Management. Besides PaaS Governance, the other main component in the architecture is the PaaS Management of Service Delivery Component (SDC) which manages the installation and configuration of the software needed to execute a service based on the recipes received from the PaaS Governance. PaaS Management manages a Products and Services Catalog that is used like repository of platform to be deployed.

Programming artifacts

The PaaS Management Platform will implement a DMTF's Cloud Infrastructure Management Interface (CIMI) (currently Work in Progress), using inside the DMTF’s Open Virtualization Format (OVF) with extensions to manage PaaS description to the upper layer. PaaS Management Platform also implements CIMI interface using also inside OVF specification of virtual appliances. Both of them are bindings in REST/HTTP.

Technologies used

Due to the PaaS Managment Platform is work in progress, we do not have the specific architecture of the PaaS Governance component.

PaaS Management Component (SDC) is developed using J2EE and installed on Apache Tomcat. Libraries needed to work with is provided by this distribution and the technologies used are:

- J2EE 5.0
- Spring 3.0.5.Release
- Jersey 1.6

Runtime pre-requisites

The PaaS Management Platform is been built and tested on Debian Squeeze (Debian GNU/Linux 6.0, kernel 2.6.32-30).

Known software requirements

- Apache Tomcat >= 7.0.12
- PostgreSQL >= 8.4
- Ruby >= 1.9.2
- RubyGem >= 1.3.5
- Chef >= 0.1.0.4
- JDK >= 1.5

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) [2] Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Due to the PaaS Managment Platform is work in progress, we do not have the specific architecture of the PaaS Governance component. For more information about the 4CaaSt project go to [1]

References

[1] http://4caast.morfeo-project.org/documentation

OpenStack Swift

Brief Description

OpenStack Swift is a “highly available, distributed, eventually consistent object/blob store [1]”. In Swift, data (objects) are stored in buckets (containers of objects). Those objects are stored with user and provider specified metadata into logical (e.g. hierarchical filesystem or tagging) and/or physical groupings (e.g. location). Objects are stored in Containers and those containers are associated with an account. Users are associated with accounts. It should be noted that Swift is not a traditional POSIX-type file system and has a little notion of directory structures.

Swift can be integrated into the Single Sign On service, Keystone.

IPR

OpenStack Swift is licensed [13] under the Apache 2.0 license [3].

Publically Available Documentation

- Overview [2]
- OpenStack Swift Administration Guide [3]
- OpenStack Swift Developer and Architecture Documentation [1]
- OpenStack Swift API [4]
- OpenStack Swift Source [5]

References

[1] http://swift.openstack.org/
[2] http://www.openstack.org/projects/storage/
[3] http://docs.openstack.org/diablo/openstack-object-storage/admin/content/
[4] http://docs.openstack.org/api/openstack-object-storage/1.0/content/
[5] https://github.com/openstack/swift

Nada Management Framework

Brief Description

The Nada Management Framework (NMF) is a set of client/server modules used to remotely configure an Application to run on virtual host running on a set of virtualized gateways. It was developed partially for the EU Nada project.

Programming artifacts

The NMF provides an API to remotely a automates the deployment of an Application distributed on a set of gateways. Those API are accessible through XML/RPC methods.

Technologies used

The virtualization technology used is Xen. The communications between modules are done through XML/RPC methods. A MySql data base is used to store basic status of the elements used in the framework.

Runtime pre-requisites

The NMF runs in an Linux environment with a Ruby 1.9.2 environment.

IPR

NMF has no IPR associated to it.

Publicly available documentation

Public information about NMF is available at http:/ / www. nanodatacenters. eu/ index. php?option=com_phocadownload& view=category& download=8:combined-deliverable-d1. 1-system-design-and-decomposition-and-d3. 1-draft-architecture-specification-of-security-privacy-and-incentive-mechanisms&id=1:architecture&Itemid=66.

OpenStack Keystone

Brief Description

OpenStack Keystone is an identity service (OpenStack Identity API [1]) that provides Single Sign On (SSO) capabilities, covering both Authentication (AuthN) and RBAC Authorisation (AuthZ).

Supported Frontend Authentication:

- HTTP Basic Authentication [2]
- OpenId [3]
- Token-based [4]

There are also plans to support storage backends of OAuth and SAML amongst others.

Supported Backend Authentication Stores:

- LDAP [5]
- SQL Databases [6] supported by SQLAlchemy [7]
- memcached [8]

There are also plans to support storage backends of PAM and ActiveDirectory amongst others.

IPR

OpenStack Keystone is licensed [13] under the Apache 2.0 license [3].

Publically Available

- Overview [9]
- OpenStack Keystone Administration Guide [10]
- OpenStack Keystone Developer and Architecture Documentation [11]
- OpenStack Keystone API [12]
- OpenStack Keystone Source [13]

References

[1] https://github.com/openstack/identity-api
[2] https://github.com/openstack/keystone/blob/master/keystone/middleware/auth_basic.py
[3] https://github.com/openstack/keystone/blob/master/keystone/middleware/auth_openid.py
[4] https://github.com/openstack/keystone/blob/master/keystone/middleware/auth_token.py
[5] https://github.com/openstack/keystone/tree/master/keystone/backends/ldap
[6] https://github.com/openstack/keystone/tree/master/keystone/backends/sqlalchemy
[7] http://www.sqlalchemy.org/docs/core/engines.html#supported-databases
[8] https://github.com/openstack/keystone/tree/master/keystone/backends/memcache
[9] http://wiki.openstack.org/keystone
[10] http://docs.openstack.org/diablo/openstack-identity/admin/content/
[11] http://keystone.openstack.org/
[12] http://docs.openstack.org/api/openstack-identity-service/2.0/content/
[13] https://github.com/openstack/keystone

Materializing Data/Context Management in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Data/Context Management chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how Data/Context Management functions are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

Publish/Subscribe Broker

Baseline Assets

- Context Awareness Platform (CAP) is TI's context management and brokerage platform based on the producer-consumer model.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.EPIC.Data.PublishSubscribe.LanguageOptimisationForMobile
- FIWARE.EPIC.Data.PublishSubscribe.Multiprotocol
- FIWARE.EPIC.Data.PublishSubscribe.RDFSupport

Features

- FIWARE.EPIC.Data.PublishSubscribe.UpdateOperation
- FIWARE.EPIC.Data.PublishSubscribe.QueryLanguage
- FIWARE.EPIC.Data.PublishSubscribe.LanguageExtentionsForOtherApplicationDomains
- FIWARE.EPIC.Data.PublishSubscribe.Subsciprion&Notification
- FIWARE.EPIC.Data.PublishSubscribe.QueryLanguageEngine
- FIWARE.EPIC.Data.PublishSubscribe.MatchingRules
- FIWARE.EPIC.Data.PublishSubscribe.MultiEntityResolution

User-Stories

- FIWARE.STORY.Data.PublishSubscribe.RDFSupport.SPARQLRestInterface
- FIWARE.STORY.Data.PublishSubscribe.RDFSupport.RDFContextInfoModeling

Complex Event Processing

Baseline Assets

- AMIT - IBM Active Middleware Technology™ is a complex event processing (CEP) engine. It aims to ease the cost of changing business logic, automating and monitoring business processes, and enabling an on demand business environment.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.Epic.Data.CEP.InboundConnectivity
- FIWARE.Epic.Data.CEP.OutboundConnectivity
- FIWARE.Epic.Data.CEP.BigDataInteroperability

BigData Analysis

Baseline Assets

- Hadoop is a heterogeneous MapReduce platform that is mainly use for ad-hoc data exploration of large sets of data.
- MongoDB is a scalable, high-performance, open source, document-oriented database.
- SAMSON Platform is a high-performance streaming MapReduce platform that is used for the near-real time analysis of streaming data.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.Epic.Data.BigData-Analysis.AlternativeLanguage
- FIWARE.Epic.Data.BigData-Analysis.API
- FIWARE.Epic.Data.BigData-Analysis.CEPLanguageInterface
- FIWARE.Epic.Data.BigData-Analysis.MapReduceExtensions
- FIWARE.Epic.Data.BigData-Analysis.ProcessCEPEvents
- FIWARE.Epic.Data.BigData-Analysis.Redundancy
- FIWARE.Epic.Data.BigData-Analysis.SharedLibrary

Features

- FIWARE.Feature.Data.BigData-Analysis.Streaming

Multimedia Analysis

Baseline Assets

- codoan (Siemens-internal asset) is a set of tools for analyzing video stream in the compressed domain.
- White Diamond is TI's platform recognizing which is the object or situation is whithin a multimedia content (picture) and collect information about this item or object or person and attached it to the content isleft employing the 'SAnr' [1] TI solution.

Epics

- FI-WARE.Epic.Data.MultimediaAnalisys.ImageCategoryDetection
- FI-WARE.Epic.Data.MultimediaAnalysis.ObjectDetection
- FI-WARE.Epic.Data.MultimediaAnalysis.ObjectTracking

User-Stories

- FI-WARE.Story.Data.MultimediaAnalysis.Configuration
- FI-WARE.Story.Data.MultimediaAnalysis.RegisterObserver
- FI-WARE.Story.Data.MultimediaAnalysis.ReleaseObserver
- FI-WARE.Story.Data.MultimediaAnalysis.EventDetectionNotification

Unstructured Data Analysis

The Unstructured Data Analysis Generic Enabler will be based on the outcomings of the FIRST (FP7-257928) project, a European Commission FP7 funded Specific Targeted Research Project started on October 1st 2010, specifically in the FIRST Analytical Pipeline. A high level description of this pipeline can be found in:

- FIRST Analytical Pipeline high level description [2]

FIRST Analytical Pipeline is composed of Data Acquisition Pipeline and Information Extraction Pipeline . More information about both pipelines can be found in their public project deliverables:

- FIRST Data Acquisition Pipeline description [3]
- FIRST Information Extraction Pipeline description [4]

Finally, more information about the integration and interactions between both pipelines can be found at:

- Data Acquisition and Information Extraction integration [5]

Meta-data Pre-processing

Baseline Assets

- Meta-data Processor is a meta-data pre-processor for RTP-encapsulated meta-data streams.

Epics

- FI-WARE.Epic.Data.MetadataPreprocessing.Configuration

User-Stories

- FI-WARE.Story.Data.MetadataPreprocessing.ConnectSource
- FI-WARE.Story.Data.MetadataPreprocessing.DisconnectSource
- FI-WARE.Story.Data.MetadataPreprocessing.ConnectSink
- FI-WARE.Story.Data.MetadataPreprocessing.DisconnectSink

Location Platform

Baseline Assets

- The location server (LOC_S) is Thales Alenia Space (TAS) scalable, high-performance, high reliability platform dedicated to location management on wireless network (2G, 2,5G, 3G, 4G).
- Location Provider (LP) is location platform of TI based on the concept of location broker allowing to connect and handle different location technologies.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FI-WARE.Epic.Data.LocalizationPlatform.AccessMngt
- FI-WARE.Epic.Data.LocalizationPlatform.QoPMngt
- FI-WARE.Epic.Data.LocalizationPlatform.PDEs_Pfolio
- FI-WARE.Epic.Data.LocalizationPlatform.EventsMngt
- FI-WARE.Epic.Data.LocatizationPlatform.RegisterLocTech
- FI-WARE.Epic.Data.LocalizationPlatform.ConfigLocPriority
- FI-WARE.Epic.Data.LocalizationPlatform.Announce&Update2PSGE
- FI-WARE.Epic.Data.LocalizationPlatform.LocSecEnforcement
- FI-WARE.Epic.Data.LocalizationPlatform.LocPropagation

Query Broker

Baseline Assets

- THESEUS QueryBroker (Siemens-internal asset) is a middleware component for unified access to distributed and heterogeneous repositories (with extensions supporting multimedia repositories).

Epics

- FI-WARE.Epic.Data.QueryBroker.API
- FI-WARE.Epic.Data.QueryBroker.QueryAccess
- FI-WARE.Epic.Data.QueryBroker.NoSQL
- FI-WARE.Epic.Data.QueryBroker.QL-Families

Features

- FI-WARE.Feature.Data.QueryBroker.FederatedQuery
- FI-WARE.Feature.Data.QueryBroker.OptimizedResultRanking
- FI-WARE.Feature.Data.QueryBroker.QueryExecutionScripting
- FI-WARE.Feature.Data.QueryBroker.PartialNULLResults

User-Stories

- FI-WARE.Story.Data.QueryBroker.RudimentaryFlickrAdapter

Semantic Annotation

Baseline Assets

- Semantic Annotator (SAnr) is TI's platform that collects the meta-data and context information about a multimedia content and attach all available information about the content to the content itself as a semantic information.

Epics

- FI-WARE.EPIC.Data.SemanticAnnotation.TagExtraction
- FI-WARE.EPIC.Data.SemanticAnnotation.OntologyDomainsConnection
- FI-WARE.EPIC.Data.SemanticAnnotation.SPARQL

Features

- FI-WARE.EPIC.Data.SemanticAnnotation.LanguageExtraction
- FI-WARE.EPIC.Data.SemanticAnnotation.KeywordsExtraction
- FI-WARE.EPIC.Data.SemanticAnnotation.ContentCategorization
- FI-WARE.EPIC.Data.SemanticAnnotation.LinkedOpenData
- FI-WARE.EPIC.Data.SemanticAnnotation.OpenOntology

Semantic Application Support

Baseline Assets

- NeOn Toolkit
- Sesame
- OWLIM

Epics

- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyEngineeringManagement
- FIWARE.EPIC.Data.SemanticApplicationSupport.ApplicationWorkspaces
- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyRegistry

Features

- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyManagement
- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyMetadataManagement
- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyVersioning
- FIWARE.EPIC.Data.SemanticApplicationSupport.Ontology&MetadataStorage
- FIWARE.EPIC.Data.SemanticApplicationSupport.OntologyRegistryInterface
- FIWARE.EPIC.Data.SemanticApplicationSupport.AutomaticOntologyPublication

Topics still being addressed at high-level

Epics

- FIWARE.Epic.Data.General.Auditing
- FIWARE.Epic.Data.General.Entities
- FIWARE.Epic.Data.General.Monitoring
- FIWARE.Epic.Data.General.NonRepudiation
- FIWARE.Epic.Data.General.Ownership
- FIWARE.Epic.Data.General.Privacy-Control
- FIWARE.Epic.Data.General.QualityHandling
- FIWARE.Epic.Data.General.Trustworthiness
- FIWARE.Epic.Data.General.CEPEnrichment
- FIWARE.Epic.Data.General.BigDataToCEP
- FIWARE.Epic.Data.General.CEPToBigData

References

[1] https://forge.fi-ware.eu/docman/view.php/9/503/TI+Semantic+Sep+2011.pdf
[2] http://project-first.eu/content/first-analytical-pipeline
[3] http://project-first.eu/sites/project-first.eu/files/content-uploads/publications/FIRST_D3.1_Semantic%20resources%20and%20data%20acquisition_v1.0_0.pdf
[4] http://project-first.eu/sites/project-first.eu/files/content-uploads/publications/FIRST_D4.1_First%20semantic%20information%20extraction%20prototype_v1.0.pdf
[5] http://project-first.eu/sites/project-first.eu/files/content-uploads/publications/FIRST_D2.2_Conceptual%20and%20technical%20integrated%20architecture%20design_v1.0.pdf

Context Awareness Platform (CAP)

Brief Description

'Context Awareness Platform' [1] involves a Context Broker (CB) and other actors playing two different roles Context Provider/Source (CP/CS) or Context Consumer (CC) or both. The main purpose is to capture the context from the context providers and sources, to represent it within a single context representation formalism and to pass it to its respective context consumers through two different mechanisms: synchronous and asynchronous. Moreover, the context broker handles the context providers advertisements in order to make then availble in the system, and subscriptions from the context consumers in order to provide the context to consumers on certain conditions.

Programming artifacts

In order to communicate with the CAP for both a new Context Provider/Source registration or for access to the context information from a Context Consumer a unique formal data rapresentation is defined called ContextML, which is based on well-known XMl or ContextQL (CQL) for more sophisticated requests or subscribtions. These requests shall be done in REST-like manner.

Technologies used

The SW is implemented as Java (JavaBeans) scripts code running in JBoss Appplication Server

Runtime pre-requisites

- Java Run-Time 6 and later
- JBean/J2EE support

Known software requirements

- JBoss AS

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program. Licensing of the software under an Open Source license is currently under consideration.

Publicly available documentation

'Context Awareness Platform architecture and formatism presentation' [1]

References

[1] https://forge.fi-ware.eu/docman/view.php/9/502/PublishSubscribe+Asset+TI+x+FI-WARE+no+hidden.pdf

AMIT

Brief Description

AMIT, IBM Active Middleware Technology™, is a lightweight and agile complex event processing (CEP) engine. It aims to ease the cost of changing business logic, automating and monitoring business processes, and enabling an on demand business environment. Our unique CEP capabilities were recognized by Gartner as an emerging market in which "enterprises will achieve new levels of flexibility and a deeper understanding of their business processes by applying the techniques of complex event processing to their daily work".

IBM Active Middleware Technology™ is capable of detecting complex situations (a context-sensitive composition of messages and events), rather than single events. The functionality is applicable when processing a composition of event sources from a business, application, or infrastructure perspective within different contexts. Sample scenarios relating to SLA alerts and compliance checking. IBM Active Middleware Technology™ uses Websphere brokering technology, implemented through processing nodes. With the use of IBM Active Middleware Technology™, the nodes monitor the message flow, enabling the detection of, and reaction to, situations that may be defined over multiple messages.

IBM Active Middleware Technology™ lets you configure the middleware in terms of business needs, thus triggering more sophisticated conclusions and enhancing automation and efficiency. IBM Active Middleware Technology™ can be configured to monitor situations relating to specific rules/events. For example, you can pre-set and automatically correct parameters for breaches of Service Level Agreements.

The added capacities of IBM Active Middleware Technology™ enhance the ability to develop advanced solutions for many domains such as the financial market, transport and logistics, banking, retail and more.

Programming artifacts

AMiT has an authoring tool which is an Eclipse based user interface for defining rules. The authoring tool is business user oriented, in which the user fills in forms with no need for code writing. There is also an option to send definitions to the AMIT engine directly using api and xml based data.

Technologies used

AMIT has a stand alone java version and a version that runs on IBM Websphere Application Server. The AMIT authoring tool is an Eclipse based java application.

Runtime pre-requisites

The AMIT stand alone version requires Java, while the other AMIT version requires the IBM Websphere Application Server.

The AMIT authoring tool requires Eclipse.

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Technology Description [1]

AMiT Documents [2]

Related Patents [3]

Demos [4]

Q&A [5]

Related Publication: Amit - the situation manager, A. Adi and O. Etzion, VLDB Journal, 2004

References

[1] https://www.research.ibm.com/haifa/dept/services/soms_ebs_tech.html
[2] https://www.research.ibm.com/haifa/dept/services/soms_ebs_documents.html
[3] https://www.research.ibm.com/haifa/dept/services/soms_ebs_patents.html
[4] https://www.research.ibm.com/haifa/dept/services/soms_ebs_demos.html
[5] https://www.research.ibm.com/haifa/dept/services/soms_ebs_qa.html

Hadoop

Brief Description

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets (> 0.5 Petabytes) across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. In particular we will be making use of Hadoop Distributed File System (HDFS) to provide a distributed storage platform that allows for data exploration, via Hive [1], as well as a data repository for large data sets and the outputs from data processing.

IPR

This product is licensed under the open source Apache License, Version 2.0 [3]

Publicly available documentation

- http://hadoop.apache.org/
- http://hadoop.apache.org/hdfs/
- http://en.wikipedia.org/wiki/Hadoop

References

[1] http://hive.apache.org/

MongoDB

Brief Description

MongoDB is an open source, high-performance, schema-free, document-oriented database written in the C++ programming language. Suited to medium sized data sets (100 Gigabytes - 500 Gigabytes), MongoDB will be used to provide a high-availability store for querying data.

IPR

This product is licensed under the open source GNU Affero General Public License, Version 3 [1]. Client interfaces are licensed under Apache License, Version 2.0 [3].

Publicly available documentation

- http://www.mongodb.org/
- http://www.mongodb.org/display/DOCS/Home
- http://en.wikipedia.org/wiki/MongoDB

References

[1] http://www.gnu.org/licenses/agpl-3.0.html

SAMSON Platform

Brief Description

SAMSON is a high-performance MapReduce platform that is designed for scaling tasks on a cluster of commodity hardware. This platform removes the need to develop a system for parallelizing and synchronizing work between servers, allowing a person to develop the analytical models needed to solve both current and future problems.

SAMSON extends the traditional MapReduce paradigm by allowing an external data source to stream data into and out of the cluster. This allows the system to handle any volume of data thus providing the ability to continuously process new inputs and output states without restriction.

Programming artifacts

The SAMSON platform uses modules to extend it's functionality. A developer generates a template file (plain text) that contains

- definition of the data-types being handled
- operational declarations, i.e. map, parse, reduce etc. that define the input and output data types
- high level script of operations needed to complete a given task

This file is then parsed and generates the necessary stubs needed to program, in C++, the operations needed to complete a task.

Technologies used

The SAMSON platform is extended using modules as documented above. Currently only C++ can be used to program the logic needed to complete a task. There is an EPIC to add an alternative language for programming. When installing SAMSON using either RPM or DEB packages, the required libraries for building modules will be provided either in the supplied packages or via the dependencies built-in to the package. Libraries needed to process data within a particular problem domain, say geo-spatial, will need to be supplied by the developer.

Runtime pre-requisites

The SAMSON platform has been built and tested on Ubuntu 11.04 and Redhat Enterprise Server 6.0. In addition to the SAMSON platform software repository, Redhat Linux will require the use of the Extra Packages for Enterprise Linux [1] repository to satisfy some dependencies. Once activated, the package manager will download all the required dependencies needed to satisfy the install process.

Known software requirements

- mongodb >= 2.0.0

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program. Licensing of the software under an Open Source license is currently under consideration.

Publicly available documentation

- SAMSON Platform High Level Overview [2]
- SAMSON Platform Architecture and operation [3]

References

[1] http://fedoraproject.org/wiki/EPEL
[2] https://forge.fi-ware.eu/docman/view.php/9/509/SAMSON_Platform.pdf
[3] https://forge.fi-ware.eu/docman/view.php/9/587/SAMSON_architecture.ppsx

Codoan

Brief description

Overall Goal:

codoan (Siemens-internal asset) is a set of tools for analyzing video stream in the compressed domain.

Approach / Features:

- Avoidance of costly video content decoding
- Video stream processing by analyzing compressed or just partially decoded syntax elements
- Very fast analysis due to hierarchical architecture

Programming artifacts

The API currently is proprietary; a RESTful API that permits easy integration into web services or other components is envisioned.

Technologies used

The codoan tool set uses network data transport protocols like RTP and RTSP. Furthermore, decoders for common video codecs, e.g., H.264/AVC or MPEG-4 Visual, are integrated in the tool stack.

Runtime pre-requisites

The tool is platform-independent and can be run either on Windows or Linux distributions. The programming language is C++.

IPR

This asset will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Some description of the asset can be found in the Product Vision of the "Meta-data Pre-processing" Generic Enabler.

White Diamond

Brief Description

'White Diamond' [1] (WD) is a platform (and Web API) for detecting & recognizing objects (items) within a multimedia content (picture) and provide additional information/resources about these items (objects, persons, covers, barcodes, etc), also leveraging TI's 'SAnr' [1] platform.

Programming artifacts

In order to communicate with the WD platform for submitting query images and retrieving information a Web API is exposed over HTTP. XML and JSON formats are available as output. Available requests are

- augment() for sending an image or a reference (URL) to be processed, which returns a query id. Metadata (such as title, context information, etc) can be added to help processing the query.
- getInfo() for retrieving (synchronously) information given a query id. A state machine information & an etag are provided to inform the requestor about the current state of the query processing

Technologies used

The SW is implemented in Java on the core (processing) part, and in PHP for the API. A common database is used (MySQL).

Runtime pre-requisites

- Java Run-Time 6 and later

Known software requirements

- LAMP stack (Apache+PHP+MySQL) for API module
- Java for Core module

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program. Licensing of the software under an Open Source license is currently under consideration.

Publicly available documentation

'White Diamond Platform architecture and functional logic presentation' [1]

References

[1] https://forge.fi-ware.eu/docman/view.php/9/504/TI+Recognition+Sep+2011.pdf

Meta-data Processor

Brief Description

Overall Goal:

The Meta-data Processor is a meta-data pre-processor for RTP-encapsulated meta-data streams.

Approach / Features:

- Timed meta-data is received and transformed.
- The input stream is encapsulated in the Real-time Transport Protocol (RTP).
- The input data is transformed into serialized Java classes using XSLT.

Programming artifacts

The API currently is proprietary; a RESTful API that permits easy integration into web services or other components is envisioned.

Technologies used

The asset is implemented in Java. The Java-internal XSLT processor is used in this asset. Version 1.6 of the JDK is required.

Runtime pre-requisites

Version 6 of the Java Runtime Environment is required.

IPR

This asset will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Some description of the asset can be found in the Product Vision of the "Meta-data Pre-processing" Generic Enabler.

Location server

Brief Description

Overall Goal:

The Location Server (LOC_S) is Thales Alenia Space (TAS) scalable, high-performance, high reliability platform dedicated to location management on wireless network (2G, 2,5G, 3G, 4G). This platform is based on various positioning techniques such as A-GPS, WiFi and Cell-Id whilst taking into account the end-user privacy

Approach / Features:

- GPS/SBAS assistance to Supl Enabled Terminals (SET)
- Location retrieval of SET using AGPS, Cell-ID and WiFi cell
- Location retrieval of non-SET via core network interfaces (CGI)
- Seamless management of multiple positioning modules
- Advanced tracking features
- Based on OMA MLP and SUPL standards & 3GPP standards

Programming artifacts

Interfaces available for third party through MLP interface

Technologies used

The asset is implemented in Java.

Runtime pre-requisites

- Java run time environment
- Alcatel Lucent middleware: Applicative Server Runtime 5350

IPR

This asset will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Some description of the asset can be found in the Product Vision of the "Location Platform"Generic Enabler.

Location Provider

Brief Description

'Location Provider' [1] is a complex Context Provider dedicated for Entities Location connected to the Context Broker (CAP) or could be connected to Publish/Subscriber GE. This components attaches many different location technologies such as: GPS, xPS, MPS (Network Based Location), NIMBLE (proprietary TI solution based on cellID and signal triangolation), WiFi and BT. It is configured for a static list of the preferred location technology from the available ones and automatically deserved by TI's Advanced User Profile (AUP) through the Context Broker (CAP) for autonomlous names (or entities) resolution (e.g. when a person by a name or login name is determined by her/hos mobile terminal identified by IMEI).

Programming artifacts

There is no programming interfaces or ways to access to this components other then the access throught the Context Broker (CAP) or, potentially in the future, Publish/Subscibe GE in FI-WARE.

Technologies used

The SW is implemented as Java (JavaBeans) scripts code running in JBoss Appplication Server

Runtime pre-requisites

- Java Run-Time 6 and later
- JBean/J2EE support

Known software requirements

- JBoss AS

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program. Licensing of the software under an Open Source license is currently under consideration.

Publicly available documentation

'Location Provider or Location Broker' [1]

References

[1] https://forge.fi-ware.eu/docman/view.php/9/524/Location+GE.pptx

QueryBroker

Brief Description

Overall Goal:

The THESEUS QueryBroker is a middleware component for unified query access to distributed and heterogeneous data repositories.

Approach / Features:

- Mapping to internal, unified representation format(s): MPEG Query Format as initial implementation for e.g. XPath, XQuery, SQL- like queries
- Synchronous/Asynchronous mode
- Timeout functionality
- Middleware based architecture (for connecting an arbitrary amount of backends)
- Loose coupled, modular architecture (easy extensibility)

Programming artifacts

The communication with the QueryBroker GE is done (currently) by valid MPQF queries (e.g by SOAP), including (de-)registration of backends with their capability descriptions and the service discovery for the distribution of queries, submitting search requests, and receiving the result sets with a requested structure.

In order to incorporate a new data repository two steps are necessary:

1. Construct a MPQF service description an store it as XML file. A complete description of the used elements can be found in the MPQF standardization document (ISO_IEC_FDIS_15938-12) along with the needed classification scheme. Within this the ServiceID points to the Java class of the according interpreter implementation (see next point).

2. Implementation of a corresponding interpreter: A MPQF interpreter in the QueryBroker GE must implement the "de.uop.dimis.air.searchInterface.SearchInterface" interface. If the original query consists of more than one query type, this initial query will be split into several sub-queries (one for each query type). This ensures, that every retrieval service only gets the query type for processing, which is defined in the corresponding service description file. Therefore, a retrieval service must translate the information encapsulated in the incoming query into the underlying language/API calls. At the end of processing, the retrieved data must be encapsulated into a valid MPQF response.

Technologies used

At its core, the Query Broker utilizes the MPQF as common internal representation for input and output query description and managing the backend search services (a comprehensive overview can be found in the paper of 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).

The asset itself is implemented in Java.

Runtime pre-requisites

- Java Run-Time 6 and later

IPR

This asset will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Some description of the asset can be found in the Product Vision of the "Query Broker" Generic Enabler.

Semantic Annotator (SAnr)

Brief Description

'Semantic Annotator (SAnr)' [1] is a platfom that analyses text to provide semantic information related to that text in terms of Linked Open Data (LOD). In particular, SANr is a multilingual tool that provides (disambiguated) LOD references for the entities mentioned in the text, together with additional information whenever available.

Programming artifacts

In order to communicate with SANr for submitting text and retrieving information a Web API is exposed over HTT, which returns JSON data. The available request is extract_words(), which provides a bag of terms, each of them being provided with a list of candidate LOD resources and additional information. A preferred disambiguated LDO candidate is indicated whenever available. Additional generic meta-information (such as language detected, etc) is provided.

Technologies used

The SW is implemented in PHP running on a LAMP platform.

Runtime pre-requisites

- N/A

Known software requirements:

- LAMP stack (Apache+PHP+MySQL)
- FreeLing POS Tagger for natural language processing

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program. Licensing of the software under an Open Source license is currently under consideration.

Publicly available documentation

'Semantic Annotator (SAnr) architecture and functional logic presentation' [1]

NeOn Toolkit

Brief Description

The NeOn Toolkit is a framework for networked ontologies development using the NeOn Methodology. The NeOn Methodology is a methodology for collaboratively building contextualized ontology networks in continuous evolution. It follows a scenario-based approach due to there are several ways to develop ontology and ontology networks and provides useful guidelines for selecting an appropriate life cycle for a specific ontology engineering project, for activities and processes evolved in ontology development, for reusing and reengineering existing ontologies or other knowledge aware resources, etc. Thus, NeOn Toolkit supports NeOn Methodology by providing tools that empowers several steps involved in NeOn Methodology like ontology editing, ontology development scheduling and so on.

NeOn Toolkit is composed of:

- A Core containing basic ontology engineering functionality. This core is open source multi-platform ontology engineering environment, which provides comprehensive support for the ontology engineering life-cycle.
- A set of plugins providing extended functionality covering a variety of ontology engineering activities, including Annotation and Documentation, Development, Human-Ontology Interaction, Knowledge Acquisition, Management, Modularization and Customization, Ontology Dynamics, Ontology Evaluation, Ontology Matching, Reasoning and Inference, and Reuse.

Programming artifacts

NeOn Toolkit allows developers to extend framework functionalities by providing plugins. This plugins can be integrated into the framework:

- Using NeOnToolkit specific extension points.
- Using general Eclipse plugin integration mechanism.

More information about plugin development can be found in public documentation section.

Technologies used

NeOn Toolkit core platform is written in Java and integrated into Eclipse using OSGi [1].

NeOn Plugins can be developed:

- In Java if they are integrated using NeOn Toolkit extension points.
- In several programming languages if they are integrated using OSGi.

Runtime pre-requisites

As NeOn Toolkit is built on top of Eclipse [2], it needs a Sun Java 6 platform to run.

IPR

NeOn Toolkit is released in several bundles, including several plugins, distributed under different licenses, so NeOn Toolkit license will change from one bundle to another. However, the basic configuration containing the core framework plus a few plugins to support basic modeling of F-logic and OWL ontologies is delivered under the Eclipse Public License (EPL [1]).

Publicly available documentation

- NeOn Toolkit 2.3 documentation [3]
- List of available NeOn Toolkit plugins [4]
- NeOn Toolkit documentation for plugin developers [5]
- NeOn Methodology for networked ontology development book [6]
- NeOn Fundation, an organization supporting NeOn, home page [7]

References

[1] http://www.osgi.org/Main/HomePage
[2] http://www.eclipse.org/
[3] http://neon-toolkit.org/w/images/Doku_2.3.pdf
[4] http://neon-toolkit.org/wiki/Neon_Plugins
[5] http://neon-toolkit.org/wiki/Developer_Corner
[6] http://www.neon-project.org/nw/NeOn_Book
[7] http://www.neon-foundation.org/

Sesame

Brief Description

Sesame is an opensource framework for RDF data storage, inference and querying. It provides means for storing RDF data, querying with SPARQL and (by default) offers RDFS reasoning

Programming artifacts

Developers can interact with the Sesame server by means of:

- a Sesame programmatic Java API.
- an HTTP communication protocol (REST) fully compliant with the SPARQL Protocol for RDF W3C Recommendation.

Developers can also customize repository behaviour by providing their own implementation of SAIL module implementing the SAIL API.

Technologies used

Sesame is implemented using Java.

Runtime pre-requisites

The Sesame server software requires the following software:

- Java 5 platform or newer.
- A Java Servlet Container that supports Java Servlet API 2.4 and Java Server Pages (JSP) 2.0, or newer.

IPR

Sesame is distributed under a BSD-style license.

Publicly available documentation

- W3C Semantic Web Activity [1]
- W3C RDF recomendations [2]
- SPARQL [3] and SPARQL 1.1 [4] RDF protocols
- Sesame User Guide [5]
- Sesame System Documentation [6]
- Sesame programatic java API [7]

References

[1] http://www.w3.org/2001/sw/#rdf
[2] http://www.w3.org/standards/techs/rdf#w3c_all
[3] http://www.w3.org/TR/rdf-sparql-protocol/
[4] http://www.w3.org/TR/sparql11-protocol/
[5] http://www.openrdf.org/doc/sesame2/users/
[6] http://www.openrdf.org/doc/sesame2/system/
[7] http://www.openrdf.org/doc/sesame2/api/

OWLIM

Brief Description

OWLIM is a SAIL API (Storage And Inference Layer) implementation written in Java that enhance Sesame capabilities by adding OWL RL support

Programming artifacts

AS OWLIM is bound to the data and query standards supported by Sesame, it supports same standards as Sesame does:

- RDF for data representation
- SeRQL and SPARQL 1.1 for querying

Technologies used

OWLIM is fully developed using Java

Runtime pre-requisites

- Java JRE version 1.5 or newer. If custom rule-sets are used then a Java JDK version 1.6 is required
- A running Sesame installation

IPR

OWLIM is available in three editions with different license agreements:

- OWLIM Lite is available free of charge under the following license [1]
- OWLIM SE is available under an RDBMS-like commercial license on a per-server-CPU basis. A free copy can be obtained for research purpose
- OWLIM Enterprise is available under an RDBMS-like commercial license on a per-server-CPU basis. A free copy can be obtained for research purpose

Publicly available documentation

- W3C OWL Working Group home page[2]
- OWL 2 Overview[3]
- OWLIM SE documentation[4]
- OWLIM Enterprise documentation (replication cluster)[5]
- SAIL API documentation [6]

References

[1] http://www.ontotext.com/owlim-lite-license-agreement
[2] http://owlim.ontotext.com/display/OWLIMv42/Home
[3] http://www.w3.org/TR/owl-overview/
[4] http://owlim.ontotext.com/display/OWLIMv42/OWLIM-SE
[5] http://owlim.ontotext.com/display/OWLIMv42/OWLIM-Enterprise
[6] http://www.openrdf.org/doc/sesame2/system/ch05.html

Materializing Internet of Things (IoT) Services Enablement in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Internet of Things (IoT) Services Enablement chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how IoT Services Enablement functions are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

IoT Communications/Connectivity Management

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Connectivity Management Generic Enabler:

- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.ConnectivityManagement

Epics

- FIWARE.Epic.IoT.ConnectivityManagement.MobilityManagement
- FIWARE.Epic.IoT.ConnectivityManagement.ConnectivityStatusManagement
- FIWARE.Epic.IoT.ConnectivityManagement.DisconnectedDeviceManagement
- FIWARE.Epic.IoT.ConnectivityManagement.SessionManagement

Features

- FIWARE.Feature.IoT.ConnectivityManagement.DisconnectedDeviceManagement.SimpleAutomatedReconnect
- FIWARE.Feature.IoT.ConnectivityManagement.DisconnectedDeviceManagement.OptimizedAutomatedReconnect
- FIWARE.Feature.IoT.ConnectivityManagement.DisconnectedDeviceManagement.TrafficCache
- FIWARE.Feature.IoT.ConnectivityManagement.SessionManagement.M2MDeviceSessionManagement
- FIWARE.Feature.IoT.ConnectivityManagement.SessionManagement.NativeDeviceSessionManagement

IoT Communications/Service Control

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Service Control Generic Enabler:

- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.ServiceControl

Epics

- FIWARE.Epic.IoT.ServiceControl.TrafficFlowManagement
- FIWARE.Epic.IoT.ServiceControl.AccessPolicyControl
- FIWARE.Epic.IoT.ServiceControl.QualityOfServiceControl
- FIWARE.Epic.IoT.ServiceControl.AddressCreation
- FIWARE.Epic.IoT.ServiceControl.AddressTranslation

IoT Communications/Device Frontend

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Devices Fronted Generic Enabler:

- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check http://www.fosstrak.org
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.DeviceFrontend

Epics

- FIWARE.Epic.IoT.DevicesFrontEnd.ProtocolAdapter
- FIWARE.Epic.IoT.DevicesFrontEnd.CommunicationProtocolAbstractionDefinition
- FIWARE.Epic.IoT.DevicesFrontEnd.NativeProtocolAdapter

Features

- FIWARE.Feature.IoT.DevicesFrontEnd.ProtocolAdapter.Codec
- FIWARE.Feature.IoT.DevicesFrontEnd.NativeProtocolAdapter.BasicReachability
- FIWARE.Feature.IoT.DevicesFrontEnd.NativeProtocolAdapter.BasicOAM
- FIWARE.Feature.IoT.DevicesFrontEnd.NativeProtocolAdapter.BasicSecurity

IoT Resource Management/Services and Resources Interaction

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Services and Resources Interaction Generic Enabler:

- IDAS is a IoT platform to automate the acquisition and management of the information retrieved from generic wireless sensor and actuator networks.
- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.
- ISIS is an infrastructure for the collection, reasoning and distribution of real world information (Things and their Attributes).
- IoT-A is the 'Internet of Things Architecture' project to establish and to evolve a federating architectural reference model for the future IoT. For more details check "http://www.iot-a.eu"
- SOL is a research program focusing on different problem domains in the area of the Internet of Things

- Sensei provides an architecture to discover and manage resources and entities of interest ("things") and interactions between them

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.ServicesAndResourcesInteraction

Epics

- FIWARE.Epic.IoT.ServicesAndResources.ResourcesAndServicesDiscovery
- FIWARE.Epic.IoT.ServicesAndResources.ResourcesDirectory
- FIWARE.Epic.IoT.ServicesAndResources.DirectoryHandler
- FIWARE.Epic.IoT.ServicesAndResources.CatalogAndLocation
- FIWARE.Epic.IoT.ServicesAndResources.AutomatedResourceCreation

Features

- FIWARE.Feature.IoT.ServicesAndResources.ResourcesAndServicesDiscovery.DiscoverServices
- FIWARE.Feature.IoT.ServicesAndResources.ResourcesAndServicesDiscovery.DiscoverResources
- FIWARE.Feature.IoT.ServicesAndResources.ResourcesDirectory.ResourceDirectory
- FIWARE.Feature.IoT.ServicesAndResources.CatalogAndLocation.GetDirectoryLocation
- FIWARE.Feature.IoT.ServicesAndResources.CatalogAndLocation.NewRegisterLocation
- FIWARE.Feature.IoT.ServicesAndResources.CatalogAndLocation.DeleteRegisterLocation
- FIWARE.Feature.IoT.ServicesAndResources.CatalogAndLocation.UpdateRegisterLocation

User-Stories

- FIWARE.Story.IoT.ServicesAndResources.ResourcesDirectory.ResourceDirectory.RegisterNewResource
- FIWARE.Story.IoT.ServicesAndResources.ResourcesDirectory.ResourceDirectory.AssociationsEntities-Resources
- FIWARE.Story.IoT.ServicesAndResources.ResourcesDirectory.ResourceDirectory.getInformationOfResource
- FIWARE.Story.IoT.ServicesAndResources.ResourcesDirectory.ResourceDirectory.getResourcesOwner
- FIWARE.Story.IoT.ServicesAndResources.ResourcesAndServicesDiscovery.DiscoverResources.Definition
- FIWARE.Story.IoT.ServicesAndResources.ResourcesAndServicesDiscovery.DiscoverResources.Development
- FIWARE.Story.IoT.ServicesAndResources.ResourcesAndServicesDiscovery.DiscoverResources.Test

IoT Resource Management/Discovery and Resolution of Things

Baseline Assets

- IoT-A is the 'Internet of Things Architecture' project to establish and to evolve a federating architectural reference model for the future IoT. For more details check "http://www.iot-a.eu"
- SOL is a research program focusing on different problem domains in the area of the Internet of Things
- Sensei provides an architecture to discover and manage resources and entities of interest ("things") and interactions between them
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.

Themes

- FIWARE.Theme.IoT.DiscoveryAndResolutionOfThings

Epics

- FIWARE.Epic.IoT.DiscoveryAndResolutionOfThings.ThingsResolution
- FIWARE.Epic.IoT.DiscoveryAndResolutionOfThings.ThingsManager
- FIWARE.Epic.IoT.DiscoveryAndResolutionOfThings.ThingsAndIoTServiceMonitoring

Features

- FIWARE.Feature.IoT.DiscoveryAndResolutionOfThings.ThingsResolution.AssociationLookUp
- FIWARE.Feature.IoT.DiscoveryAndResolutionOfThings.ThingsResolution.AssociationDiscovery
- FIWARE.Feature.IoT.DiscoveryAndResolutionOfThings.ThingsManager.AssociationManagement

User-Stories

- FIWARE.Story.IoT.DiscoveryAndResolutionOfThings.ThingsManager.RegisterNewAssociation

IoT Data Handling/Local Storage

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Local Storage Generic Enabler:

- ISIS is an infrastructure for the collection, reasoning and distribution of real world information (Things and their Attributes).
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.
- Sensei provides an architecture to discover and manage resources and entities of interest ("things") and interactions between them

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.LocalStorage

Epics

- FIWARE.Epic.IoT.LocalStorage.MicroDataBase
- FIWARE.Epic.IoT.LocalStorage.InstantaneousAndRealtimeData

IoT Data Handling/Data Pooling

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Data Pooling Generic Enabler:

- ISIS is an infrastructure for the collection, reasoning and distribution of real world information (Things and their Attributes).
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.
- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.
- Sensei provides an architecture to discover and manage resources and entities of interest ("things") and interactions between them
- SOL is a research program focusing on different problem domains in the area of the Internet of Things

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.DataPooling

Epics

- FIWARE.Epic.IoT.DataPooling.LocalDataDiscovery
- FIWARE.Epic.IoT.DataPooling.DataModelsExposure
- FIWARE.Epic.IoT.DataPooling.DataModelSemanticIntegration

IoT Data Handling/Data Access Policy

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Data Access Policy Generic Enabler:

- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.
- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.DataAccessPolicy

Epics

- FIWARE.Epic.IoT.DataAccessPolicy.AnonymousMechanisms
- FIWARE.Epic.IoT.DataAccessPolicy.AccessRightsControl

IoT Data Handling/Data Handling

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Data Handling Generic Enabler:

- ISIS is an infrastructure for the collection, reasoning and distribution of real world information (Things and their Attributes).
- FossTrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications. For more details check "http://www.fosstrak.org"
- M2MPlanet/Pangoo is a M2M research platform with user interface to deploy and monitor sensors and actuators networks, including gateways.
- Cumulocity is a cloud-enabled application and management platform for machine-to-machine services.
- Sensei provides an architecture to discover and manage resources and entities of interest ("things") and interactions between them
- SOL is a research program focusing on different problem domains in the area of the Internet of Things
- Orange CEP Application Server is an application server dedicated to complex event processing. It is typically used on top of middlewares or mediation layers, in order to propagate value-added and filtered data by aggregation or composition.
- Petals ESB is an open source Enterprise Service Bus (ESB), the service oriented infrastructure for service oriented architectures (SOA). Petals ESB is suitable for deployment in real-time, rapidly changing environments that are typically common in large scale enterprise SOA solutions. Check this url [1] for details.

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.DataHandling

Epics

- FIWARE.Epic.IoT.DataHandling.EventsGenerator
- FIWARE.Epic.IoT.DataHandling.DataFilteringAndAggregation
- FIWARE.Epic.IoT.DataHandling.IoTSubscribe
- FIWARE.Epic.IoT.DataHandling.IoTPublish
- FIWARE.Epic.IoT.DataHandling.LocalStorage

Features

- FIWARE.Feature.IoT.DataHandling.EventsGenerator.EpcGlobalEcReports
- FIWARE.Feature.IoT.DataHandling.EventsGenerator.M2mPlanet
- FIWARE.Feature.IoT.DataHandling.EventsGenerator.Isis
- FIWARE.Feature.IoT.DataHandling.EventsGenerator.MobileSensors
- FIWARE.Feature.IoT.DataHandling.EventsGenerator.Ngsi
- FIWARE.Feature.IoT.DataHandling.EventsGenerator.EventTemplate
- FIWARE.Feature.IoT.DataHandling.DataFilteringAndAggregation.Filtering
- FIWARE.Feature.IoT.DataHandling.DataFilteringAndAggregation.Aggregation
- FIWARE.Feature.IoT.DataHandling.DataFilteringAndAggregation.DeviceEmbeddedCep
- FIWARE.Feature.IoT.DataHandling.IoTSubscribe.Device
- FIWARE.Feature.IoT.DataHandling.IoTSubscribe.Broker
- FIWARE.Feature.IoT.DataHandling.IoTPublish.Device
- FIWARE.Feature.IoT.DataHandling.IoTPublish.Broker
- FIWARE.Feature.IoT.DataHandling.LocalStorage.EpcisRepository

User-Stories

- FIWARE.Story.IoT.DataHandling.EventsGenerator.EpcGlobalEcReports.ReadRfidTags
- FIWARE.Story.IoT.DataHandling.DataFilteringAndAggregation.Filtering.RfidEvent
- FIWARE.Story.IoT.DataHandling.DataFilteringAndAggregation.Aggregation.RfidEvent
- FIWARE.Story.IoT.DataHandling.IoTPublish.Broker.CepToEsb
- FIWARE.Story.IoT.DataHandling.IoTPublish.Broker.EsbToCep
- FIWARE.Story.IoT.DataHandling.LocalStorage.EpcisRepository.ObjectEvent
- FIWARE.Story.IoT.DataHandling.LocalStorage.EpcisRepository.AggregationEvent

IoT Process Automation/Template Handler

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Template Handler Generic Enabler:

- IoT-A is the 'Internet of Things Architecture' project to establish and to evolve a federating architectural reference model for the future IoT. For more details check "http://www.iot-a.eu"

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.TemplateHandler

Epics

- FIWARE.Epic.IoT.TemplateHandler.UnreliableDataHandling
- FIWARE.Epic.IoT.TemplateHandler.UnreliableResourceHandling
- FIWARE.Epic.IoT.TemplateHandler.DecentralisedProcesses
- FIWARE.Epic.IoT.TemplateHandler.EventDrivenProcesses
- FIWARE.Epic.IoT.TemplateHandler.RealTimeProcesses
- FIWARE.Epic.IoT.TemplateHandler.ThingBasedProcesses

Features

- FIWARE.Feature.IoT.TemplateHandler.ProcessCreation
- FIWARE.Feature.IoT.TemplateHandler.ProcessExecution
- FIWARE.Feature.IoT.TemplateHandler.ServiceInteroperability

User-Stories

- FIWARE.Story.IoT.TemplateHandler.ProcessCreation.TemplateCreation

IoT Process Automation/Semantic

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Semantic Generic Enabler:

- Linked Data Platform and Gateway: the linked sensor data platform supports publication and access to resource and entity descriptions described by the semantic models. The semantic descriptions, wherever applicable, are provided in association to concepts defined on the Linked Open Data cloud (e.g. location data, semantic tags, etc.).

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.Semantic

Epics

- FIWARE.Epic.IoT.Semantic.IntelligentDecisionMaking

Features

- FIWARE.Feature.IoT.Semantic.SemanticMediator
- FIWARE.Feature.IoT.Semantic.ContextCollector
- FIWARE.Feature.IoT.Semantic.ContextMediator

User-Stories

- FIWARE.Story.IoT.Semantic.SemanticMediator.OntologyDeployment

IoT Process Automation/Exposure

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the Exposure Generic Enabler:

- ISIS is an infrastructure for the collection, reasoning and distribution of real world information (Things and their Attributes).
- Linked Data Platform and Gateway the linked sensor data platform supports publication and access to resource and entity descriptions described by the semantic models. The semantic descriptions, wherever applicable, are provided in association to concepts defined on the Linked Open Data cloud (e.g. location data, semantic tags, etc.). This will provide query mechanisms to access the semantic descriptions and will help discovery and

resolution infrastructure to identify and/or select resources/services that fulfill requirements of a business process workflow.
- IoT-A is the 'Internet of Things Architecture' project to establish and to evolve a federating architectural reference model for the future IoT. For more details check "http://www.iot-a.eu"

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Themes

- FIWARE.Theme.IoT.Exposure

Epics

- FIWARE.Epic.IoT.Exposure.DeviceStatusMonitoring
- FIWARE.Epic.IoT.Exposure.DeviceCommunicationFailure
- FIWARE.Epic.IoT.Exposure.ProcessAccess
- FIWARE.Epic.IoT.Exposure.ProcessDeployment
- FIWARE.Epic.IoT.Exposure.ProcessRemoteExecution
- FIWARE.Epic.IoT.Exposure.SemanticHandler
- FIWARE.Epic.IoT.Exposure.Actuation
- FIWARE.Epic.IoT.Exposure.ThingLevelInteraction

Features

- FIWARE.Feature.IoT.Exposure.DeviceCapabilitiesDiscovery
- FIWARE.Feature.IoT.Exposure.DeviceCapabilitiesList
- FIWARE.Feature.IoT.Exposure.DeviceInformationAccess
- FIWARE.Feature.IoT.Exposure.DeviceControl
- FIWARE.Feature.IoT.Exposure.DeviceReconfigure
- FIWARE.Feature.IoT.Exposure.ThingBasedQuery
- FIWARE.Feature.IoT.Exposure.AsynchronousThingBasedInteraction

Topics still being addressed at high-level

Epics

- FIWARE.Epic.IoT.GatewaySecurityMonitoring
- FIWARE.Epic.IoT.LowPowerEncryption
- FIWARE.Epic.IoT.Devices&GatewaysAnonymization

References

[1] http://www.ow2.org/view/ActivitiesDashboard/Petals_ESB

Cumulocity

Brief Description

Cumulocity offers a cloud-enabled application and management platform for machine-to-machine services.

The application part of the platform simplifies the implementation of M2M services by hiding the complexities in communicating with the diverse devices in their own protocols and transport mechanisms. It also provides ready-made applications and a component library for rapid development of own applications.

The management part of the platform simplifies running M2M services – that is deploying, assuring and charging for the services. It configures the devices, monitors their functioning and provides charging records based on the observed service usage.

Being cloud-enabled means that users of Cumulocity are not required to own any IT infrastructure beyond network connectivity and a web browser. This also allows small and medium enterprises to provide full M2M services at low cost.

Programming artifacts

The main remote interface technology is JSON/REST/HTTPS (using authentication). This is neutral towards the client technology and requires only minimal infrastructure to develop against. The main local interface technology is OSGi Declarative Services. This decouples consumers and producers of services effectively, does not cause any performance penalty and is very lightweight.

The key features and functional components of Cumulocity M2M application platform are:

Device Integration Layer

The purpose of the device integration layer is to provide an easy way to integrate any meter or sensor to the system. It consists of following features:

- Built-in protocol support. Cumulocity supports out of the box DLMS, GE's SmartSync, Landis & Gyr command center, MultiSpeak and ThereCorp.
- O&M Agent SDK. Cumulocity has an open SDK for writing additional Agents that 3rd parties can use to enable Cumulocity to manage additional devices.
- Smart Device Integration API. Cumulocity has an open API for integrating 3rd party sensors and meters into the system. This enables third parties to integrate with Cumulocity without using the Cumulocity SDK (this means teams can use any language and tools for integration)
- Simulators. Device simulators make it testing different type exceptional conditions easy.

Cumulocity Core

Cumulocity core consists of the core business logic components as well the data stores of the system. It contains the following parts:

- Tenant Service. Cumulocity supports hosting several enterprises (tenants) on the same physical or virtual hardware. Tenant service is responsible for managing tenants as well branding aspects. Among other things it creates the tenant specific physical data stores. The tenant service is part of the Cumulocity core and tenant manager is the client application part.
- AA Service. Cumulocity uses spring security for authentication and authorization. As enablers it also contains user and group services for storing users and groups and associating these with different authorities.
- Identity Service. Identity Manager is responsible for mapping between identities that device vendors assign to their devices, identities that enterprise business processes use to manage devices and internal Cumulocity

identities. The identity service is part of Cumulocity core and Identity Manager is the client application that is part of the platform applications suite.

- Data stores (RDBMS and NoSQL). Cumulocity uses two different data store technologies: relational and document oriented data stores. They are used as follows:
  - Platform Data. Tenant Manager uses MySQL for storing information about different customers (tenants), what is their status, what applications have they currently subscribed to and so on.
  - Tenant Data stores. Each tenant have their own physically separate datastores Two technologies are used:
    - Relational Model (MySQL). MySQL is used for storing information about AA (Users, Groups and their associations with authorities) and for identity management
    - Document Oriented NoSQL(CouchDb). CouchDB is used for storing event and measurements data as well the inventory is stored in CouchDB.

Software Development Tools

- Inventory. Cumulocity contains architecture for implementing inventory for different verticals and customer solutions. A number of standard inventory implementations are part of the system:
  - Device inventory is used to store access details so that data collection and configuration management can connect to the device.
  - Topology data is also stored in the inventory. Cumulocity contains several topologies such as electric distribution hierarchy and geographical hierarchy but it can be augmented with additional hierarchies
  - Customer specific topologies for example grouping devices by organizational units. The Inventory in Cumulocity is also extensible so customer specific data and relationships can easily be added to the system.
- Complex Event Processing Engine CEP engine allows for creation of real time event driven applications. It can be used in many aspects in M2M. For data validation, aggregation of data, correlation of unrelated events and generation of new events and so on. Each vertical solution comes up with its own embedded CEP applications that contain the needed business logic. CEP engine is also available for customer specific solution development
- Rules engine for externalizing simple decisions to a human readable database that are likely to be changed at runtime (like how to prioritize trouble tickets).
- Reporting engine is used for creating different customer specific reports. Cumolocity based vertical solutions each comes up with a number of solution specific reports.
- Rapid Application Development (RAD). Cumulocity supports RAD by enabling rapid GUI development with a Rich Intenet Application (RIA) framework. The RAD support is especially useful in creating additional user interfaces that support customer specific additions to the database (such as customer specific inventory data).
- Full Text Search. Cumulocoity employs a full text search engine. It will automatically index all the data in Cumulocity NoSQL database including customer specific extensions and inventory implementations.

Exposure APIs

- Functional REST API. Cumulocity has a RESTful exposure API for northbound applications to use its functionalities.
- Batch integration API. Batch interface is used for exporting large dataset. It is used for example in billing integration where meter readings are transferred to a billing system.
- Event API. Publish/subscribe interface allows for receiving event information from a device or set of devices in near real time. This allows for creation of independent event driven applications.

Technologies used

The main data storage is based on Apache CouchDB [1]. The event processing engine is based on Esper [2]. For communicating with Cumulocity, REST over JSON [3] is used. For best integration with Cumulocity, applications should be developed as OSGi [4] bundles. Authentication and authorization for applications is based on Spring Security [5]. For developing new functionality, we use Eclipse [6].

For user interfaces, we have used two technologies:

- If you would like to develop enterprise user interfaces with more complex widgets and a more complex layout and/or if you are not so familiar with HTML, DOM and CSS, we recommend EXT JS [7].
- For more traditional web applications, we have used jQuery [8] and jQuery-UI [9] components.

Runtime pre-requisites

Cumulocity development environment for Agent and Application development requires only a Java development kit and the Eclipse IDE.

For local Cumulocity installation Ubuntu 10.04 LTS recommended, however other Debian-based Linux distributions may work. To complete the dependencies the following packages are needed: CouchDB, Java Development Kit, Tomcat, Hyperic Server and Agent.

Known software requirements

- JDK >= 1.6.0_24
- Eclipse >= 3.6.2

IPR

This product is currently closed source.

Publicly available documentation

Currently not available.

References

[1] http://couchdb.apache.org/
[2] http://esper.codehaus.org/
[3] http://www.json.org/
[4] http://www.osgi.org/Specifications/HomePage
[5] http://static.springsource.org/spring-security/site/
[6] http://www.eclipse.org/documentation/
[7] http://www.sencha.com/
[8] http://jquery.com/
[9] http://jqueryui.com/

FossTrak

Brief Description

Fosstrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications.

Fosstrak EPCIS is a complete implementation of the EPCIS standard specification (Version 1.0.1 of September 21, 2007). It successfully passed conformance testing and is EPCglobal-certified.

Fosstrak is based upon different software modules:

- EPCIS Repository: Fosstrak EPCIS provides an EPCglobal-certified EPCIS Repository as well as Query and Capture clients. In addition to these standards-compliant modules, it also offers a "Webadapter" for easy EPCIS access via web protocols (e.g., REST).
- Tag Data Translation Library: The Fosstrak TDT Engine offers and easy way to convert between different EPC representations.
- ALE Middleware: RFID tag filtering and collection layer with LLRP support
- LLRP Commander: a Graphic User Interface that allows easy LLRP interactions and configuration with LLRP compliant RFID readers

Fosstrak has been combined with Esper in order to improve its features.

Technologies used

All Fosstrak components are developed in Java.

Runtime pre-requisites

The Fosstrak stack has been run and tested on Windows XP Pro.

Known software requirements

- EPCIS Repository: JDK >= 1.6.0_05, Apache Tomcat servlet container >= 5.5, MySQL server >= 5.0
- TDT: JDK >= 1.6.0_05
- ALE Middleware: JDK >= 1.6.0_05, Apache Tomcat servlet container >= 5.5
- LLRP Commander: Java >= 1.5, Eclipse >= 3.3.0, Rifidi reader simulator

IPR

All Fosstrak components [1] are licensed under the GNU Lesser General Public License v2.1 [2] except Fosstrak LLRP Commander [3]

Fosstrak LLRP Commander [3] is licensed under the GNU General Public License v3 [4]

Publicly available documentation

- Fosstrak website [5]
- EPCIS Repository User Guide [6]
- EPCIS Repository Developer Guide [7]
- EPCIS Repository Architecture Guide [8]
- TDT Overview [9]

- TDT User Guide [10]
- ALE Middleware User Guide [11]
- ALE Middleware Developper Guide [12]
- LLRP Commander User Guide [13]

References

[1] http://www.fosstrak.org/
[2] http://www.gnu.org/licenses/lgpl-2.1-standalone.html
[3] http://www.fosstrak.org/llrp/index.html
[4] http://www.gnu.org/licenses/gpl-3.0-standalone.html
[5] http://www.fosstrak.org
[6] http://www.fosstrak.org/epcis/docs/user-guide.html
[7] http://www.fosstrak.org/epcis/docs/developer-guide.html
[8] http://www.fosstrak.org/epcis/docs/architecture-guide.html
[9] http://www.fosstrak.org/tdt/index.html
[10] http://www.fosstrak.org/tdt/user-index.html
[11] http://www.fosstrak.org/fc/docs/user-index.html
[12] http://www.fosstrak.org/fc/docs/developer-index.html
[13] http://www.fosstrak.org/llrp/docs/user-index.html

M2MPlanet/Pangoo

Brief Description

PANGOO is a fully stable platform to support sensors networks design and deployment. The network design include multiroute solutions for each sensor and identification and deployment of gateways based on radio technology constraints. Pangoo provides also monitoring facilities for users as sensors and gateways status, multi-routing failures and real-time modification.

M2M Planet is a platform component to collect and aggregate data from all sensors associated to a gateway. This component is assocaited to a relational database to store all the sensors data.

Programming artifacts

PANGOO uses dedicated parameters files to define user profiles and associated rights.Dedicated APIs are available.

M2M Planet provides some APIs but all features could be used through PANGOO APIs.

Technologies used

PANGOO API is based on Web Services (WSDL).

M2MPlanet has been developed in EJB technologies and provides Web Services APIs.

Runtime pre-requisites

Pangoo runtime does not required any specific component.

M2M Planet is based on a JONAS developer environment and does not need another specific component.

Both can run on Virtual Machines (VMWare)

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program.

IDAS

IDAS – Intelligence Data Advanced Solution (TELEFONICA I+D)

Brief Description

IDAS has been conceived as an end-to-end open platform intended to be used in a broad range of Internet-of-Things application scenarios and services. Together with providing a basic support to IoT service providers, different types of service users are envisaged, varying from 'individual end-users' to 'industrial users', automating the acquisition and management of the information retrieved from generic wireless sensor and actuator networks.

Programming artifacts

The technology provided is an API based on Webservices interface available to 3rd party applications. REST is in the future roadmap, including direct query capabilities over database through http.

Technologies used

API Webservices interface available to 3rd party applications. REST is in the future roadmap, including direct query capabilities over database through http.

Core capabilities have been coded using C++.

The platform is based on available standards:

- Service Enablers : Open Mobile Alliance (OMA) Service Environment (OSE)
- Sensor Web Enablement family of standards from Open Geospatial Consortium (OGC®): SensorML, O&M, SOS, …

The platform provides a set of APIs to easily integrate existing sensor networks into the information processing system. Another set of APIs allows the creation of services based on the capabilities offered by the platform.Telefónica's IDAS Platform provides the following northbound and southbound interfaces:

- Northbound basic functionalities using SIP native communication protocols, HTTP and Web Services technology following the OGC SWE (Sensor Web Enablement) specifications: Resource Discovery, Publish-Subscribe-Notify, Event-filtering & Processing, Homogeneous Remote Management, and Measurement Storage
- Southbound basic functions(Gateway):

Communication Protocol Adaptation: to convert from SIP communication protocol (used in the Platform) to the required communication protocol used in the Wireless Sensor Network and vice versa.

- Data format Adaptation: to translate device specific Messages into SensorML-based Messages and vice versa.
- This provides an abstraction level to the platform making it suitable for integrating different sensor technologies. So, an especific Adaptor module is needed for including new protocols.

Runtime pre-requisites

The IDAS platform has been built on Linux Redhat 5.5.

Known software requirements

- mongodb repository v1.8.1
- monit-5.2.5-linux-x64.tar
- xml-xalan-C_1_10_0
- XQilla-2.2.4
- xerces-c-3.1.1
- pion-net-3.0.23
- openssl-1.0.0c
- zeromq-2.0.10
- mongo: mongo-cxx-driver-v1.8
- boost_1_45_0
- log4cplus-1.0.4

IPR

IPR status: Closed source. The IDAS platform has been internally developed by Telefónica based on previously developed in-house technology. It is based on a generic architecture for integrating heterogeneous and disperse M2M devices, sensor & actuator technologies.TID is the owner of the platform. It will be available for being used in the Core Platform through its API. The terms of use of the IDAS would need to be further discussed and agreed.

Publicly available documentation

- IDAS Platform High Level Overview [1]

References

[1] https://forge.fi-ware.eu/docman/view.php/11/523/IDAS-Description-Fiware.pdf

ISIS

Isis – context management framework (NEC Research Lab Europe)

Brief Description

Isis is an infrastructure for the collection, reasoning and distribution of real world information, and changing the status of the real world by performing actuation. An Isis deployment typically consists of several interconnected Agents running on devices, gateways and server side. It supports the APIs to add new sensors, actuators and information processing components. Applications can talk to any agent to retrieve the information available anywhere in Isis.

Programming artifacts

Applications interact with Isis Agents by a high-level interface very similar to OMA NGSI, which means that the data model is thing/attribute based. The interface is implemented using XML-RPC.

Isis uses a plugin architecture for adding new data sources and actuators. In order to add a sensor to a system where an Isis agent runs, developers need to define a retriever using a Java Properties File. The retriever definition includes the Java class that obtains the raw sensor data (Source Class) and the class responsible for mapping this data into the thing/attribute based data model. The implementation of Actuators follows the same approach.

Technologies used

The framework is implemented in Java (using JavaSE 1.4). Applications can be programmed in any language, as the communication with the Isis Agents is via XML-RPC.

Runtime pre-requisites

As a java program having a low footprint, Isis runs on all kinds of devices from Android and Windows Mobile phones to full size servers.

IPR

IPR status: This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

IoT-A

IoT-A– Internet of Things Architecture (NEC, TID, Univ.Surrey)

Brief Description

IoT-A (Internet-of-Things Architecture), an FP7 ICT project, develops an architectural reference model together with the definition of an initial set of key building blocks. These enable the integration of IoT into the service layer of the Future Internet, and realize a novel resolution infrastructure, as well as a network infrastructure that allows the seamless communication flow between IoT devices and services. Through the implementation of real-life use cases the benefits of the developed architecture will be demonstrated.

Programming artifacts

Not defined yet.

Technologies used

Not defined yet.

Runtime pre-requisites

Not defined yet.

Known software requirements

Not defined yet.

IPR

IPR status: Closed source, although mostly not implemented.

Publicly available documentation

- IDAS website [1]

References

[1] http://www.iot-a.eu

Atos SOL

Brief Description

SOL is a research program by the Atos Research & Innovation Smart Objects Lab, focusing on different problem domains in the area of the Internet of Things. Among the domains that are being investigated are:

- Complex Event Processing
- Device Event Routing and Aggregation
- Resource Handling
- Virtual object composites in smart ecosystems (cities, buildings, etc.)

Programming artifacts

Complex Event Processing

The ongoing SOL/CEP project deals with Complex Event Processing, whereby different event types are processed and synthesized into more complex events, driven by a Specification language.

The software component under development is aimed at accepting an extensible array of input formats and generating different output formats, by means of pluggable adapters. One of the goals is the development of adapters capable of processing the data format proposed in the Data and Content Management chapter.

Resource Handling

The SOL/RES is a continuing project, aimed at organizing, grouping and associating names to physical resources, by means of a Resource Directory, so as to form a companion component towards SOL/CEP.

Currently, interaction with the Resource Directory is handled through lightweight API calls, but more advanced methods such as channel listeners and protocols are contemplated, decoupling the necessity of API-bindings.

The API supports the insertion, lookup, deletion etc. of resources, where resources are identified by their address and carry attributes such as logical name, type, and their capabilities.

Technologies used

Both components mentioned are aimed at maximum performance, for which they are conceived in C/C++.

Runtime pre-requisites

The platforms are being developed and tested on FreeBSD and Windows environments.

As far as SOL/RES is concerned, the portable nature of the code will allow the generation of libraries for any platform desired.

Known software requirements

- C/C++
- Unix / Windows

IPR

Software components are released as closed source, but APIs and protocols will be published.

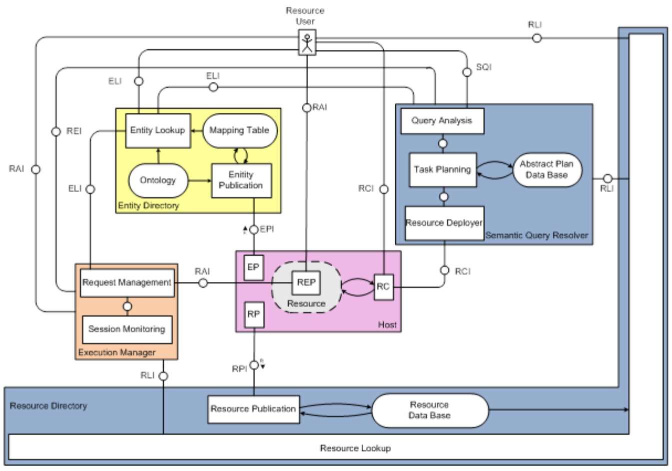
Publicly available documentation

Currently, there is no publicly available documentation. Atos is working on material to be published.

Sensei

Brief Description

The EU FP7 Project delivered an architecture (see figure below) to integrate heterogeneous wireless sensor and actuator networks with the (future) Internet. As in FI-WARE, the actual hardware is hidden behind a concept called *Resource*, which is a representation of the functionality of a certain instrument (e.g., sensor, actuator etc.).



SENSEI architecture *(©SENSEI Consortium; SENSEI Deliverable D3.6, to be appear on* http://www.sensei-project.eu/*)*

The main components of this architecture are:

- **Resource Directory** (RD): Stores the descriptions of the resources (static information), and can provide a list of resources that meet the users' criteria. The resource descriptions consist of two parts:
  - The basic resource description: This part contains information such as an identifier (ID), a name for the resource, a set of tags where the basic capabilities can be described, and also the syntactic description of the interfaces to access the resources (usually the WSDL, WADL or WIDL description of the resource).
  - The advanced resource description: This part is considered optional and includes semantic descriptions for the operations and information provided by the resource. Utilising the advanced description, some powerful functions such as dynamic resource composition offered by SENSEI components can be provided to the final users.
- **Entity Directory** (ED): Maintains the associations between the properties of the Entity of Interest (EoI, the "things") and the resources supplying information or interaction capabilities related to these EoIs.
- **Semantic Query Resolver** (SQR): Used for complex queries. The SQR is able to receive queries that contain semantic information through a declarative language, analyze the queries and find the appropriate set of resources to satisfy the query using the basic rendez-vous components (Resource and Entity Directory). If the query includes any form of aggregation of information or interaction resources, the SQR will construct a plan involving a sequence of resources. If required resources do not exist, a resource provider will attempt to create and deploy them. An example query formulated in English is "Provide the average temperature of Stockholm". The SQR will

receive this query in a SENSEI formatted request and deduce that the entity of interest is a place (Stockholm) and the desirable property of the EoI is temperature. The SQR will consult the entity directory to find out the temperature resources associated with the entity Stockholm and according to a plan of action for such types of queries (average temperature), it will compose an aggregation (averaging) tree to produce the result.
- **Execution Manager** (EM): Used to handle long-term interactions with resources, e.g., subscriptions. The EM can support sessions, events (i.e. notify the user when the temperature reaches 25℃) and more complicated functionality like the resource adaption (i.e. dynamically change the resource that provides the information back to the user due to a failure without any user interaction). Can also be used to monitor state of resources continuously by providing an appropriate execution plan.

Mapping to FI-WARE GE's

An initial mapping of functionality can be found in the following Excel Sheet [1].

Runtime pre-requisites

Known software requirements

- Windows or Linux operating system
- Java SDK, version 6.0 or higher
- Eclipse IDE for Java EE Developers
- SEDNA server [2]
- MySQL Server, version 5.0

IPR

The rights to the different components lie with the SENSEI project partner that developed the respective component. Currently source code is only available to SENSEI consortium partners. Details can be found in the SENSEI Cook Book [3]; but the IPR ownership is summarized in the following table:

Component	IPR Owner
Resource Directory	Ericsson
Entity Directory	SAP
Semantic Query Resolver	SAP
Execution Manager	NEC

Publicly available documentation

- Overwiew white paper [4]
- Developers' "cook book [5]" with detailed explanations:
- Project Web site: http://www.sensei-project.eu/(the references above can also be found from here)

References

[1] https://forge.fi-ware.eu/docman/view.php/11/532/fi_ware_iot_resources_managment_epics.xls
[2] http://www.sedna.org/download.html
[3] https://ncit-cluster.grid.pub.ro/trac/Sensei-WP5-Public/wiki/WikiStart
[4] http://www.ict-sensei.org/index.php?option=com_docman&task=doc_download&gid=83&Itemid=49
[5] https://ncit-cluster.grid.pub.ro/trac/Sensei-WP5-Public

Linked Data Platform and Gateway

Brief Description

The linked data platform enables to publish Semantic Sensor Network data and to link the data to existing resources on the Web. The platform supports publication of extensible and interoperable resource, entity and service descriptions in the form of linked data. It also supports association of different concepts from Linked Open Data cloud to annotate the resources and entities. The semantically enriched data is available at HTTP level that makes it available for business processing methods and data integration and collaboration services.

This enables users to publish their sensor description data as RDF triples, use other existing RDF description data, link the descriptions to the existing resources on publicly available linked data repositories and make it available to the data consumers through SPARQL endpoints.

The platform provides an annotation interface that helps users to create descriptions and annotate them using other resources. It uses Jena API to query DBpedia (http://dbpedia.org) and other resources to obtain annotation concepts for location, type and descriptive properties. The Linked Open Data resources are queried and the results are serialised using AJAX technology directly to the page. The platform also provides Restful interfaces which enable users/devices to publish the descriptions directly to the repository. For example, in our gateway solution (Ganz et al, http://dx.doi.org/10.1145/2016551.2016557) the resource descriptions are directly published into the repository. Figure 1. shows the main components in the platform.

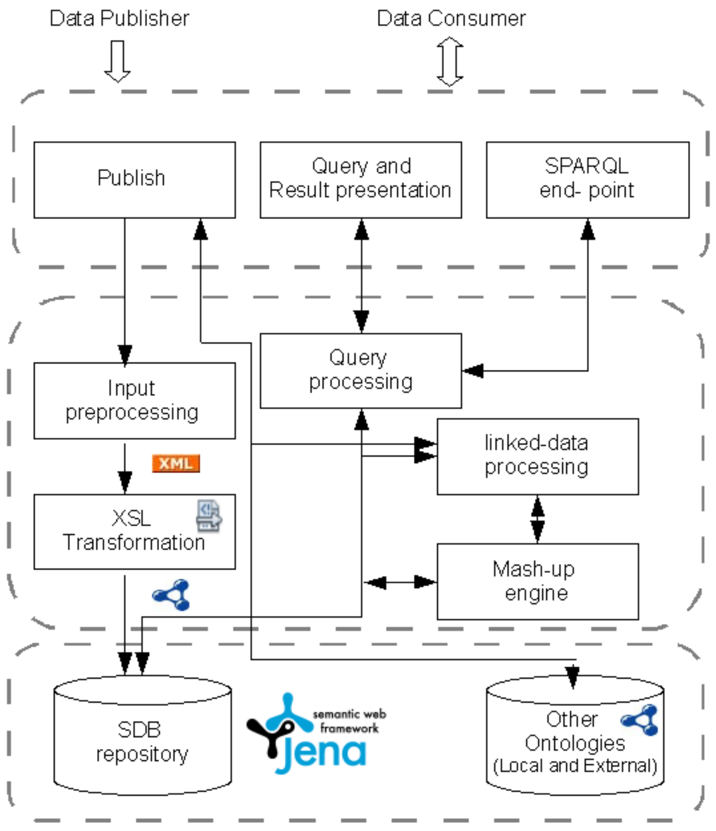


Fig. 1. Main components in the linked sensor data platform

The current version is a proof of concept implementation and uses information models that are designed in the context of the IoT-A project (the models are discussed in De et al, fedcsis.eucip.pl/proceedings/pliks/113.pdf). Publication of the data is possible using Restful interfaces or the descriptions can be created using the annotation interfaces provided in the system. The main aspects of the system are described in (Barnaghi et al, http://ceur-ws. org/Vol-668/paper2.pdf) and also in the final report of the W3C Semantic Sensor Networks Incubator group (http:/

/www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/)

An intial deployment of the system (still under test and not stable) is available at: http://iotserver3.ee.surrey.ac.uk:8080/IOTA/

Programming artifacts

It runs on an Apache TomCat server and annotation and query tools are implemented using Java Servlet and JSP technologies. Using these, it will be an HTTP level access.

We also provide RESTFUL interfaces to access the GET (description), POST (publish description), DELETE (unpublish description); SPARQL query for semantic queries can be supported using a SPARQL end-pint (Joseki) or servlet access (with SPARQL protocol support).

Technologies used

The system is developed using J2EE technologies and open source APIs. The following is a list of the tools and technologies that are used in the system.

- Jena Framework – for handling RDF data and SPARQL queries (http://jena.sourceforge.net/)
- SDB (http://openjena.org/SDB/) a SPARQL database for Jena to store the RDF triples.
- Joseki (http://joseki.sourceforge.net/), to provide SPARQL endpoints, we use an open source SPARQL server for Jena.
- Google Maps API (http://code.google.com/apis/maps/) for the demonstrator.
- DBPedia (http://dbpedia.org/) resources are queried online.
- AJAX technologies to serialise the results while querying DBPedia using the annotation tools.

Runtime pre-requisites

- Java runtime libraries (J2EE support)
- Apache TomCat server
- Clients (HTTP Browser, Servlet, or RESTFUL).

IPR

University of Surrey- apart from those APIs that are mentioned above. Currently - no plans to make it open source. TBD.

Publicly available documentation

- Barnaghi et al, http://ceur-ws.org/Vol-668/paper2.pdf
- The final report of the W3C Semantic Sensor Networks Incubator group (http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/)
- Gateway: Ganz et al, http://dx.doi.org/10.1145/2016551.2016557
- De et al, fedcsis.eucip.pl/proceedings/pliks/113.pdf

Materializing Applications/Services Ecosystem and Delivery Framework in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Apps/Services Ecosystem and Delivery chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how Apps/Services Ecosystem and Delivery functions are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

Generic Themes

The following themes are cross different enablers:

- Theme:Aggregator creates mashup
- Theme:Consumer buying service
- Theme:Multidevice/Multichannel access

USDL Service Descriptions

Baseline Assets

- USDL 3.0 Specifications M5 - A set of service description schemas that are used to describe the various aspects of services accross the service life-cycle. USDL has a focus on describing business aspects but links also to technical aspects described elsewhere. USDL can be used as a common unified description format that enables sharing information about services among the various components of the FI-Ware platform.

Epics

- FIWARE.EPIC.Apps.USDL.DefineUSDLExtension

Features

- FIWARE.FEATURE.Apps.USDL.BasicVocabulariesSpecification

User stories

- FIWARE.STORY.Apps.USDL.LDCoreSpecification
- FIWARE.STORY.Apps.USDL.LDIoTSpecification
- FIWARE.STORY.Apps.USDL.LDSECSpecification
- FIWARE.STORY.Apps.USDL.LDSLASpecification
- FIWARE.STORY.Apps.USDL.LDPricingSpecification

- FIWARE.STORY.Apps.USDL.Pricing.Requirements
- FIWARE.STORY.Apps.USDL.LDIPRSpecification
- FIWARE.STORY.Apps.USDL.LDLegalSpecification

Service Repository

Baseline Assets

- Apache Chemistry Client and Server - Apache Chemistry provides open source implementations of the Content Management Interoperability Services (CMIS) specification.
- JENA SDB/TDB RDF Store - An RDF store implementation that provides SPARQL HTTP

Epics

- FIWARE.EPIC.Apps.Repository.MaintainServiceDescription

Features

- FIWARE.FEATURE.Apps.USDL.BasicRepositoryService

User stories

- FIWARE.STORY.Apps.Repository.UploadServiceDescription
- FIWARE.STORY.Apps.Repository.RetrieveServiceDescription
- FIWARE.STORY.Apps.Repository.ListServiceDescriptions
- FIWARE.STORY.Apps.Repository.RepositorySearch

Registry

Baseline Assets

- JENA SDB/TDB RDF Store - An RDF store implementation that provides SPARQL HTTP
- MongoDB - a distributed NoSQL database for JSON objects

Epics

- FIWARE.EPIC.Apps.Registry.RegisterProviderAndStore
- FIWARE.EPIC.Apps.Registry.GetProviderAndStoreInfo

User stories

- FIWARE.STORY.Apps.Registry.LDProviderSpecification
- FIWARE.STORY.Apps.Registry.LDStoreSpecification
- FIWARE.STORY.Apps.Registry.RegisterProvider
- FIWARE.STORY.Apps.Registry.RegisterStore

Marketplace

Baseline Assets

- USDL Service Marketplace OS - A service marketplace implementation out of the Thesesus/TEXO project.
- PREMIUM-Services Pricing Strategies Simulator - A simulation-based tool to support a service provider's pricing decisions

Epics

- FIWARE.EPIC.Apps.Marketplace.ManageStoresAndOffers
- FIWARE.EPIC.Apps.Marketplace.PricingSupport
- FIWARE.EPIC.Apps.Marketplace.RatingAndFeedback
- FIWARE.EPIC.Apps.Marketplace.ServiceComparison
- FIWARE.EPIC.Apps.Marketplace.ServiceDiscovery

Features

- FIWARE.FEATURE.Apps.Marketplace.BasicMarketplaceServices

User Stories

- FIWARE.STORY.Apps.Marketplace.OfferingAPI
- FIWARE.STORY.Apps.Marketplace.StoreAPI
- FIWARE.STORY.Apps.Marketplace.RatingAPI
- FIWARE.STORY.Apps.Marketplace.ReviewAPI
- FIWARE.STORY.Apps.Marketplace.SearchAPI
- FIWARE.STORY.Apps.Marketplace.DiscoveryAPI
- FIWARE.STORY.Apps.Marketplace.ComparisonAPIn
- FIWARE.STORY.Apps.Marketplace.AnalyticsAPI
- FIWARE.STORY.Apps.Marketplace.MonitoringAPI

Store

Baseline Assets

- Ericsson eStore - A service store that handles the final business transaction (buying) and the whole related back office process.

Epics

- FIWARE.EPIC.Apps.Store.ServiceConfiguration
- FIWARE.EPIC.Apps.Store.Contracting
- FIWARE.EPIC.Apps.Store.ManageServiceOffers
- FIWARE.EPIC.Apps.Store.CompletionManagement

Business Elements & Business Models

Baseline Assets

No baseline asset available, it will be addressed under the scope of a future FI-WARE open call.

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.EPIC.Apps.BEBM.MaintainBusinessModels&Elements
- FIWARE.EPIC.Apps.BEBM.MaintainServicesAppsActors

Revenue Sharing

Baseline Assets

This asset is based on the Revenue Sharing product developed by Telefónica I+D as part of the Bussiness Framework initiative.

- Revenue Sharing System

Following is the compilation of entries in the Backlog followed for materialization of this Generic Enabler.

Epics

- FIWARE.EPIC.Apps.RSSS.CDRSDRLoading
- FIWARE.EPIC.Apps.RSSS.DataAggregation
- FIWARE.EPIC.Apps.RSSS.RSCalculation
- FIWARE.EPIC.Apps.RSSS.PaymentGeneration

Features

- FIWARE.FEATURE.Apps.RSSS.FlexibleRSSAlgorithms

User-Stories

- FIWARE.STORY.Apps.RSSS.FlexibleRSSAlgorithmProvision
- FIWARE.STORY.Apps.RSSS.FlexibleRSSAlgorithmUsage
- FIWARE.STORY.Apps.RSSS.DefineExpRSSmodel
- FIWARE.STORY.Apps.RSSS.DefineLinearCostRSSmodel

Composition

Baseline Assets

Following is a list of the main assets that have been adopted as baseline for the reference implementation of the Composition Editor and Composition Execution Engine generic enablers.

- EricssonCommonCompositionCore is a tool set to create and execute composed services and applications. It contains an event- and constraint-based composition editor and a dynamic late-binding service composition engine.
- WebConnectivityEnabler is an application level communication protocol providing it's own addressing and messaging semantics. In composition it can be used for enabling communication (wiring) among web gadgets in different web contexts.

- Wirecloud Mashup Platform is a framework that allows end-users to mashup/compose and execute service front-ends (also known as gadgets/widgets). It consists of an application mashup editor, a mashup execution engine and a gadget catalogue.
- Mashup Factory is a toolset to compose own services and applications. A graphical editor supports the combination of services in a data flow oriented mash-up style, later to be deployed on an appropriate orchestration engine.
- LighSemantic-enabled Design Time Semi-automatic Service Composition is a suite for creating processes as service compositions at design time using standard modelling languages and light semantic descriptions and annotations.

Epics

- FIWARE.EPIC.Apps.CompositionEditor.CheckApp
- FIWARE.EPIC.Apps.CompositionEditor.ChooseServiceAtShop
- FIWARE.EPIC.Apps.CompositionEditor.CompareOffers
- FIWARE.EPIC.Apps.CompositionEditor.ConfigureService
- FIWARE.EPIC.Apps.CompositionEditor.CreateComposition
- FIWARE.EPIC.Apps.CompositionEditor.CreateServiceDescription
- FIWARE.EPIC.Apps.CompositionEditor.DeployService
- FIWARE.EPIC.Apps.CompositionEditor.DeterminePrice
- FIWARE.EPIC.Apps.CompositionEditor.GetContracts
- FIWARE.EPIC.Apps.CompositionEditor.PublishService
- FIWARE.EPIC.Apps.CompositionEditor.SearchServiceOffer
- FIWARE.EPIC.Apps.CompositionEditor.UtilizeIoTBPMNExtensionsForComposition
- FIWARE.EPIC.Apps.ExecutionEngine.ExecuteComposedService

Features

- FIWARE.FEATURE.Apps.CompositionEditor.C3BasicSearchUSDLService
- FIWARE.FEATURE.Apps.ExecutionEngine.C3MakeExecutionEngineCloudReady
- FIWARE.FEATURE.Apps.CompositionEditor.IoTBPMNExtensions
- FIWARE.FEATURE.Apps.ExecutionEngine.SupportEventDrivenGadgetExecution
- FIWARE.FEATURE.Apps.CompositionEditor.CreateUIOrientedServiceComposition
- FIWARE.FEATURE.Apps.ExecutionEngine.SupportPushDrivenWiring

User-Stories

- FIWARE.STORY.Apps.CompositionEditor.CreateEventBasedComposition
- FIWARE.STORY.Apps.ExecutionEngine.C3CreateLoadBalancedExecution
- FIWARE.STORY.Apps.ExecutionEngine.C3IdentifyandParseUSDL
- FIWARE.STORY.Apps.ExecutionEngine.SupportPushDrivenGadgets
- FIWARE.STORY.Apps.CompositionEditor.CreateMashupFromCatalog
- FIWARE.STORY.Apps.CompositionEditor.ExportMashupDefinition
- FIWARE.STORY.Apps.CompositionEditor.ImportMashup
- FIWARE.STORY.Apps.CompositionEditor.AnnotateServiceComposition

Mediation

Baseline Assets

- WSO2-Mediation-Layer-Extension - Extension of an OS framework WSO2 that provides mediation capabilities
- Light-Semantics-enable Process and Data Mediation Composer

Epics

- FIWARE.EPIC.Apps.Gateway.Aggregator.Repository.DomainSpecificExtensions
- FIWARE.EPIC.Apps.Mediation.DataMediation
- FIWARE.EPIC.Apps.Mediation.ProtocolMediation

User-Stories

- FIWARE.STORY.Apps.ProtocolMediation.POX2SOAPMediation
- FIWARE.STORY.Apps.ProtocolMediation.HTTP2JMSTransportSwitching
- FIWARE.STORY.Apps.Gateway.Mediator.ServiceTaskBinding
- FIWARE.STORY.Apps.Gateway.Mediator.ServiceTaskExpansion
- FIWARE.STORY.Apps.Gateway.Mediator.ServiceCompositionDataMapping

SLA Management

Baseline Assets

Following is the list of main assets that have been adopted as baseline for the reference implementation of the SLA Management Generic Enabler.

- SLA Model : A model of Service Level Agreement content and semantics.

Following is the compilation of entries in the Backlog followed for materialisation of this Generic Enabler.

Epics

- FIWARE.EPIC.Apps.SLAManager.SLATemplateDesign
- FIWARE.EPIC.Apps.SLAManager.SLATemplateDiscovery
- FIWARE.EPIC.Apps.SLAManager.SLANegotiation
- FIWARE.EPIC.Apps.SLAManager.SLAMonitoring

Features

- FIWARE.FEATURE.Apps.SLAManager.SLAModel

Multi-channel/Multi-device Access System

Baseline Assets

No baseline asset available, it will be addressed under the scope of a future FI-WARE open call.

Epics

- FIWARE.EPIC.Apps.MultiChannel-Device.MultiDeviceAccess
- FIWARE.EPIC.Apps.MultiChannel-Device.MultiChannelAccess

Topics still being addressed at high-level

Epics

- FIWARE.EPIC.Apps.OpenIssues.SecurityAspects
- FIWARE.EPIC.Apps.OpenIssues.ChannelMaker
- FIWARE.EPIC.Apps.OpenIssues.Hoster

User stories

- FIWARE.STORY.Apps.ChannelMaker.DefineAPI2Devices
- FIWARE.STORY.Apps.Hoster.DefineServices

USDL 3.0 Specifications M5

Brief description

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. SAP Research believes that 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.

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) and infrastructure (e.g. CPU and storage services). A generic service description language – like USDL – acting as a “one size fits all” 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 and its USDL Application meta-model.

In FI-Ware we will rely on the expressiveness of USDL version M5. Instead of using the specialized XML serialization of USDL M5 we will switch over to a Linked Data Representation of USDL. By maintaining the expressiveness of USDL we will ...

A wide variety of use cases and perspectives were investigated across the different publicly funded research projects to develop service descriptions for forms of services and an alignment of business and technical aspects not available through previous R&D and standards efforts.

Use cases from the corporate world shed insights into commercial management and arrangements of services such as cost centre ownership and provisioning, releasing and dependencies in complex IT and business landscapes. Use cases involving service marketplaces procuring services as complex as those from SAP’s portfolio and ecosystem provided new insights into structures for service bundling including price-competitive of both professional and automated forms of services. Use cases from cloud computing/IT virtualization helped frame platform and infrastructure services into USDL and extended service dependency graph with hosting requirements for services. Use cases from business networks have shown that service versioning/provisioning capabilities need to extend beyond service providers to intermediaries and outsourced players such as brokers, aggregators and channel partners – to drive up “network effect” of services.

IPR

The M5 specifications of USDL contain a dedicated terms of use section, where the IPR are defined.

Publicly available documentation

The projects have included: the TEXO project within the THESEUS pogram initiated by the German Federal Ministry of Economics and Technology (BMWi), German Federal Ministry of Education and Research projects (BMBF) Premium Services, EU European Commission, DG INFSO projects FAST [1], RESERVOIR [1], MASTER [2], SERVFACE [3], SHAPE [4], SLO@SOI [5], and SOA4ALL [6] and finally the Australian Smart Services GRC [7].

Further description and specifications for USDL can be found on [8].

References

[1] http://fast.morfeo-project.eu/
[2] http://www.master-fp7.eu/
[3] http://www.servface.org/
[4] http://www.shape-project.eu/
[5] http://sla-at-soi.eu/
[6] http://www.soa4all.eu/
[7] http://www.smartservicescrc.com.au/default.html
[8] http://internet-of-services.com/index.php?id=264

Apache Chemistry Client and Server

Brief Description

Apache Chemistry is a client and server implementation for the Content Management Interoperability Services (CMIS) specification.

From Wikipedia:

Content Management Interoperability Services (CMIS) is a specification for improving interoperability between Enterprise Content Management systems. OASIS, a web standards consortium, approved CMIS as an OASIS Specification on May 1, 2010.

CMIS provides a common data model covering typed files, folders with generic properties that can be set or read. In addition there may be an access control system, and a checkout and version control facility, and the ability to define generic relations. There is a set of generic services for modifying and querying the data model, and several protocol bindings for these services, including SOAP and Representational State Transfer (REST), using the Atom convention. The model is based on common architectures of document management systems.

Although initiated by AIIM, CMIS is now being administered by the OASIS standards body. Participants in the process include Adobe Systems Incorporated, Alfresco, EMC, eXo, FatWire, HP, IBM, ISIS Papyrus, Liferay, Microsoft, Open Text, Oracle and SAP. The standard is available for public comment at OASIS.

IPR

- Apache Software License [1]

Publicly available documentation

- Apache Chemistry Web Site [2]
- Content Management Interoperability Services (CMIS) [3]
- Wikipedia: Content Management Interoperability Services [4]

References

[1] http://www.apache.org/licenses/
[2] http://chemistry.apache.org/
[3] http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html
[4] http://en.wikipedia.org/wiki/Content_Management_Interoperability_Services

JENA SDB/TDB RDF Store

Brief Description

From the Website:

"SDB is a component of Jena. It provides for scalable storage and query of RDF datasets using conventional SQL databases for use in standalone applications, J2EE and other application frameworks. The database tools for load balancing, security, clustering, backup and administration can all be used to manage the installation. SDB is designed specifically to support SPARQL, the query language developed by the W3C RDF Data Access Working Group."

IPR

- SDB License [1]

Jena includes software developed by the Apache Software Foundation (http://www.apache.org/).

Publicly available documentation

- A database for Jena [2]
- OpenJena Website [3]

References

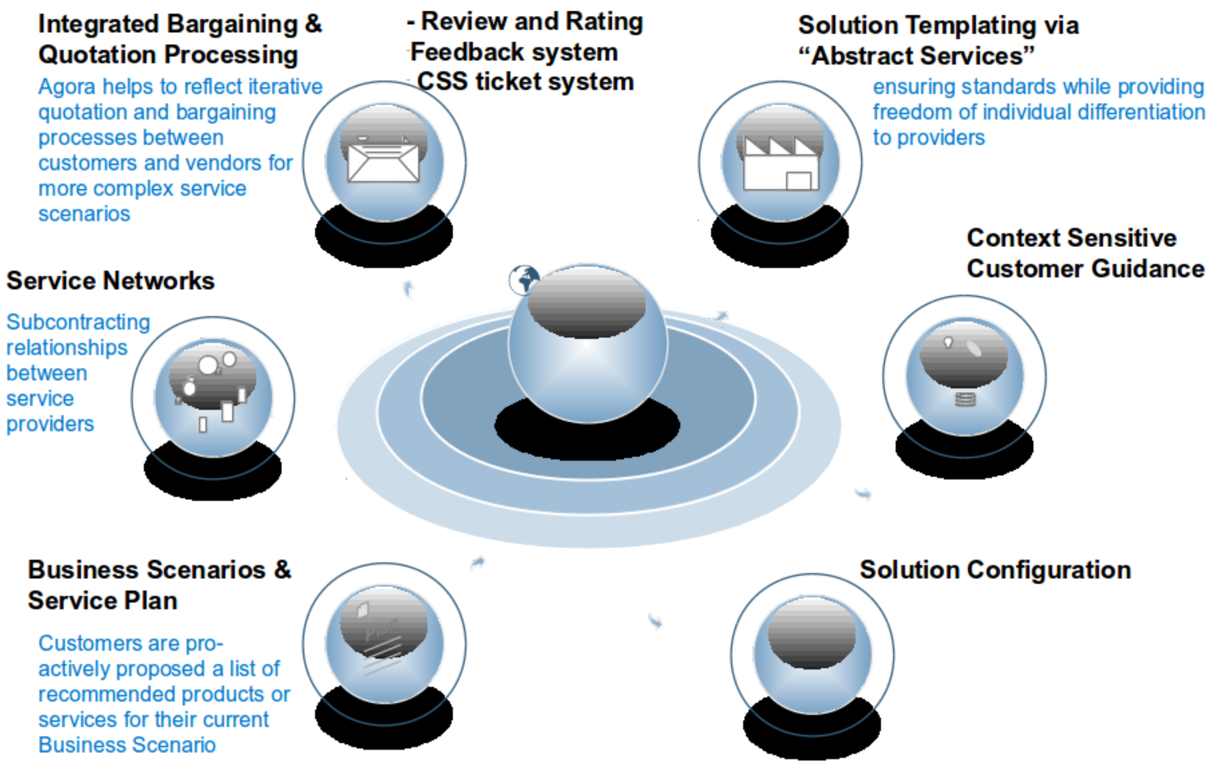
[1] http://openjena.org/SDB/license.html
[2] http://openjena.org/SDB/
[3] http://openjena.org

USDL Service Marketplace OS

Brief Description

Internet based business networks require a marketplace, where you can offer and deal with services like goods and finally combine them to value added services. On the marketplace you can quickly find and book services and automated services, which enables you to attend an industry-ecosystem better than before.

The USDL service marketplace enables the Internet of Services to realize the following vision: Services become tradable goods which can get 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 gets extended. USDL service marketplace as essential place to go for finding services and solutions Solutions of different partners can get combined Individual service offers are possible for any customer Service partners offer own and sub-arranged services simultaneously


Integrated Bargaining & Quotation Processing
Agora helps to reflect iterative quotation and bargaining processes between customers and vendors for more complex service scenarios
- Review and Rating Feedback system CSS ticket system
Solution Templating via "Abstract Services"
ensuring standards while providing freedom of individual differentiation to providers
Service Networks
Subcontracting relationships between service providers
Context Sensitive Customer Guidance
Business Scenarios & Service Plan
Customers are pro-actively proposed a list of recommended products or services for their current Business Scenario
Solution Configuration

The marketplace includes all functionalities for service trading including the topics on the following picture:

Programming artifacts

The marketplace is embedded in the TEXO tool chain. It covers a number of adjustable processes for the customer, intermediaries and vendors. The customer can identify his/her demand, search and match existing services on the market, browse catalogs, do pricing calculations (including VAT and tax handling), configure the service offering, request a final price, order the service, monitor the ordering and installation status, and finally give feedback. The intermediaries can do administration, service maintenance, control contracting and commisioning.

The vendors can display customer request, create special offers and bargaining, receive and process orders, update the status, and invoke billing. For FI-Ware the marketplace functionality will be reduced because the high-level architecture separates marketplace and shop functions at least conceptually. Therefore the interfaces for publishing offerings, registering shops, authentication, request for quotation and other marketplace functions will be wrapped by dedicated FI-Ware GE APIs.

Technologies used

The following technologies are used in the repository asset:

- J2EE,
- Jboss IDE, AS, Hibernate, Seam richfaces
- ANTL
- Apache
- Hibernate
- Spring
- SSL and X.509 certificates
- Postgres, MaxDB
- OpenCSV
- Linux (Ubuntu)

The relies on the Texo USDL repository and some functionality (probably not necessary for FI-Ware) relies on additional SAP products and TEXO partner contributions.

IPR

Open Source is in planning

Publicly available documentation

The marketplace was used and further developed in the projects TEXO, Premium Services.

- Internet of Services Marketplace [1]
- Premium Services Homepage [2]

For FI-Ware not all marketplace functionality is needed, because the functionality is split-up into marketplace and store functionality. The challenge is to refactor the system in the appropriate way.

References

[1] http://www.internet-of-services.com/index.php?id=277&L=elzjeyql
[2] http://premiumservices.research-events.com/joomla/

PREMIUM-Services Pricing Strategies Simulator

Brief Description

The asset 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, 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.

Programming artifacts

The asset consists of a standalone web application that allows the user to:

- create a new market scenario or load an existing one from an Excel file
- define an experiment in terms of price levels or price strategies to be tested
- visualize the results of the experiments graphically and numerically to select the best price level or strategy

Technologies used

- J2EE for the backend simulation engine
- JQuery for the user interface
- Excel as an import format for market scenarii

Runtime pre-requisites

The asset runs on a Java Application Server (it has been tested on Glassfish 2.1.1 and JBoss 5).

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

- Vision for the asset [1]
- PREMIUM|Services homepage [2]

References

[1] https://forge.fi-ware.eu/docman/view.php/12/525/2011+-+Novelli+-+An+Agent+Based+Approach+To+Strategic+CVM.pdf
[2] http://www.premium-services.de/

Ericsson eStore

Name of the Asset

Ericsson eStore

Description of the concept

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. 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.

Programming artifacts

The eStore offering currently provides the following two main features to the end user: • Application Store - A site, which enables end users to in an easy manner discover, purchase and download applications, widgets and services. The application store can filter and present applications based on meta-data (categories, for example), user device, user profile, purchase history, and user search queries. The site also supports user rating of applications and users subscribing to content for a time period. • Personal Page - A browser-based service, which enables end users to see and control applications they have bought or subscribed to, and downloaded, as well as control, their server side services and widgets.

Maturity/Estimated efforts for the service deployment

While eStore is an existing product, it needs significant modifications to integrate it with USDL and the FIWARE vision.

Technologies Used

Jetty Oracle Linux Virtual Server CentOs Sun Solaris

Pre-requisites

Jetty Oracle Linux Virtual Server CentOs Sun Solaris

IPR

The product is going to be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to the prerequisites of the FI-PPP program.

Revenue Sharing System

Brief description

This asset is involved in the post-processing of the sale of a good (service/application) to a customer. The customer pays for the good an amount of money which has to be shared between the different actors involved: provider and marketplace/store owner. The Revenue Sharing system receives and collects service/call data records from the different billing systems involved and calculates the appropriate revenue for the provider and marketplace/store owner. It is not available to end users.

Programming artifacts

This asset consists of a standalone product and application, featuring several components that offer the full functionality. As such it does not offer any programming interfaces or APIs for developers to extend it.

Technologies Used

The following technologies are used in this asset:

- Adobe Flex 3, Php 5.3 (Zend 2.3), HTML 4, JQuery, AXIS 1.4, Struts 2.1
- J2EE 5
- Oracle 10.2.0.4
- Apache Tomcat 6
- RED HAT ENTERPRISE LINUX 4.5

Runtime pre-requisites

Monitoring and accounting information is not provided by this component. Billing and other kind of systems provide charging/billing information to this system.

IPR

All rights held by Telefónica. This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

Related projects:

- 4CAAST [1]
- BlueVia [1]

References

[1] https://bluevia.com/en/

EricssonCommonCompositionCore

Name of the Asset

Ericsson Common Composition Core (C3)

Description of the concept

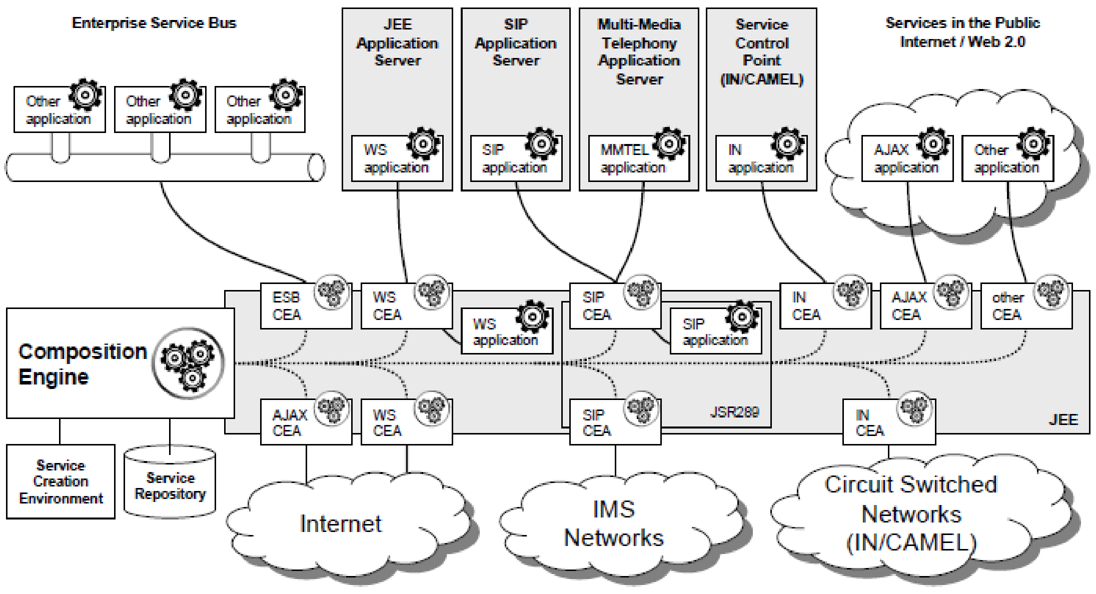
The Ericsson Common Composition Core (C3) is a toolset for subject matter experts, business professionals or end user prosumers to create and execute composed services and applications. The main characteristics of the C3is that it a) enables composition of business logic driven by asynchronous events (which is well suited for communication services that are event, rather than process driven), and b) uses a constraint-based decision process with late-binding of services in the composition resolution process. The C3 consists of a Composition Editor and a Service Composition Engine.

Description of the service

The Composition Editor offers a graphical user interface (GUI) to construct and configure composed services and applications. The editor allows the creation of composed service skeletons. The skeletons provide the business logic, the data and control flow, and service placeholders. 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. First class support for events is provided. External and internal events may start actions, events can be filtered, events can be thrown, and scopes can be defined. The editor has facilities for consistency checking of interfaces and simple debugging. It connects to the execution engine to allow testing, debugging, deploying, executing, controlling and post execution analysis of the composed applications.

The Composition 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. Using dynamic late-binding composition, the engine creates a workflow on the fly from a matching skeleton. It then it executes the business logic and manages the dataflow transformation and the control flow elements specified in the skeleton step-by-step. The process is triggered by a composition execution agent (CEA) that receives a triggering event and requests the next step from the composition engine. At each step the relevant services for the skeleton execution are chosen 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.

Execution agents exist for SIP, WS, REST, WARP, RMI, JBI, and IoT interfaces.



Programming artifacts

The Composition Editor is using a UI to interface with the user in the composition creation process. The composition Engine is interfacing underlying orchestration engines via a uniform interface.

Maturity/Estimated efforts for the service deployment

The maturity level is high, the C3 is developed towards an Ericsson product.

Technologies Used

Sailfin, OpenDS, SIP, HTTP, REST, SOAP, AOP, Comet, Netty, LDAP, Mongo, JBI, EJB, Javascript, Jruby, Groovy, Scala, Linux, Esper/Complex Event Processing, Infinispan, Hazelcast, JMS

Pre-requisites

Sailfin, OpenDS

IPR

The product is going to be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to the prerequisites of the FI-PPP program.

WebConnectivityEnabler

Name of the Asset

Web Connectivity Enabler (WARP)

Description of the concept

Warp is an application level protocol providing it's own addressing and messaging semantics. The key features of Warp are unified addressing - a fully symmetric fine-grained RESTful mechanism that eliminates client/server distinction and enables overall node reachability; bidirectional asynchronous messaging that enables efficient and secure interaction between loosely coupled entities over diverse network transports; and flexible routing that can be seamlessly integrated with value added services.

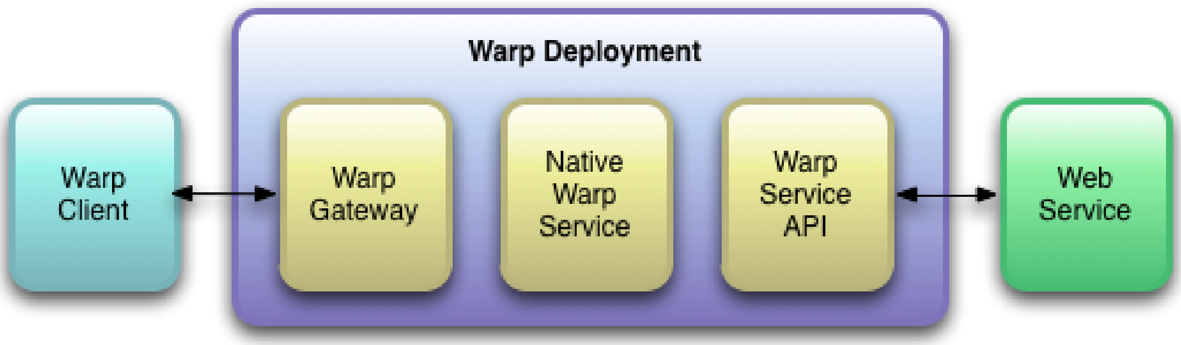
Description of the service

WARP is an overlay network providing it's own addressing and messaging semantics. The key requirements on the Warp overlay network are addressability, reachability and asynchronous messaging. Addressability is important both on an application and component level; it allows entities on the network to know more precisely who they are talking to other than a given "host". Reachability and asynchronous messaging provides both clients and servers with the ability to initiate conversation to each-other without waiting for one or the other to take an initiative. For example, a server shall be able to update a client's status without the client requesting a status update.

While every entity on the Warp overlay network should be able to treat any other entity as equal in terms of addressing and messaging, there are different ways to make such equality possible. In Warp we identify two different kinds of entities that need to communicate: Services, always-on network-based entities, and Clients, devices or applications that are potentially temporary and/or sporadically available. A Service is manually configured, does not change its state often, and is accessible without having to maintain an underlying network connection. Examples of Clients are mobile phones (which may lose network connectivity), or computer applications (which need to maintain a TCP connection open in order to be reachable).

In order to solve the issue of addressability and reachability, a Warp Client connects to a Warp Gateway. By doing so, the Warp Client is assigned an address in the Warp overlay network, and thus becomes addressable in it. The Warp Client and the Warp Gateway maintain their underlying transport connection alive and capable of transporting Warp messages that are destined to the Warp Client. Once the Warp Client is addressable, it can engage in mutual communication with the other Warp Clients or Services through the Warp Gateway.

Services, in turn, come in two different varieties. There are Native Warp Services, which are developed and deployed in such a manner that they communicate using Warp messages directly. These Services can access other Services and Warp Clients immediately, but they need to be deployed and configured with Warp in mind, in a special deployment scenario. Therefore, the Warp Service API is available, making it possible to expose any Internet-accessible Web Service as a Warp Service, with a few minor or no modifications to the Web Service. The Warp Service API allows any Web Service, using an API key, to send messages to other Warp Services or Warp Clients. It also makes the Web Service addressable via Warp, and allows the Web Service to receive Warp messages.



Programming artifacts

There are 4 core WARP modules: Lookup Service, Router, Gateway, Authentication Provider that provide APIs to access their respective functions. A Javascript client library will provide a common API to be accesible from within the browser.

Maturity/Estimated efforts for the service deployment

Warp is a mature research prototype that is developed for commercial deployments.

Technologies Used

- Glassfish Java Application Server
- Jetty (open source HTTP client/server library)
- EclipseLink (an implementation of JPA)
- MySQL database
- HQSQL database
- OpenEJB (open source embeddable and lightweight EJB Container System)
- SSL and X.509 certificates
- Jersey (implementation of JAX-RS specification)
- Bouncy Castle (a collection of APIs used in cryptography)

Pre-requisites

See above technologies.

IPR

The product is going to be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to the prerequisites of the FI-PPP program.

Wirecloud Mashup Platform

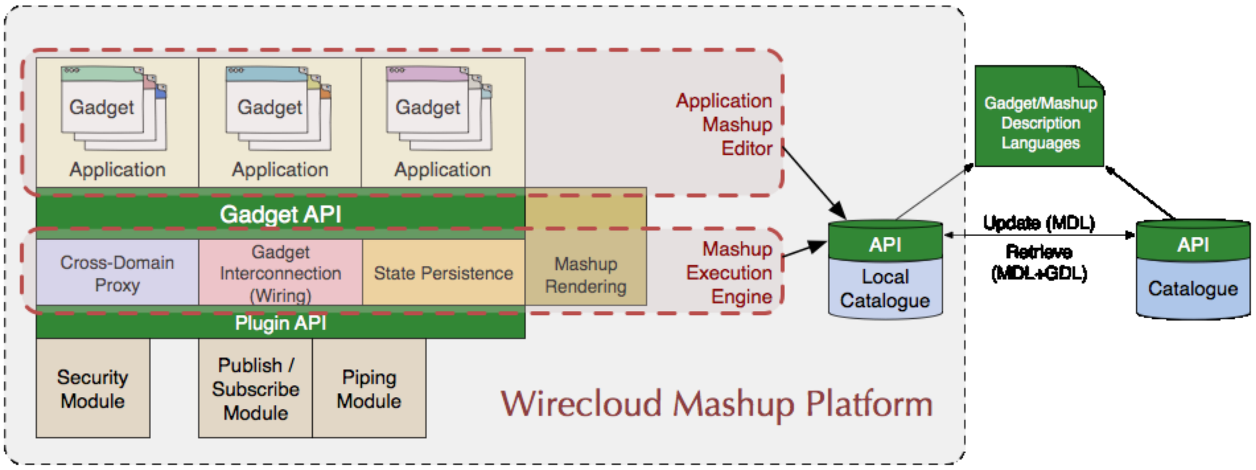
Brief Description

The Wirecloud Mashup Platform is a framework that allows end-users to mashup/compose service front-ends (also known as gadgets/widgets) in order to create their own composite applications. Gadgets are the building blocks of the interface and behaviour that constitute the single access elements to the underlying back-end services. These gadgets are usually provided by 3rd parties through a catalogue, and they can be visually interconnected through a graphical editor by means of a process called Wiring.

Programming artifacts

The Wirecloud Mashup Platform is made up of the following components:

- Application Mashup Editor
- Mashup Execution Engine
- Catalogue



The **Application Mashup Editor** conforms the web-based composition editor that end-users utilize to create their mashup applications. It consists of a workspace where end-users can place gadgets in a spatial manner, a wiring tool to set the interconnection between gadgets, and a catalogue/marketplace where end-users can access and look for the gadgets they need.

The **Mashup Execution Engine** offers gadget interconnection, mashup state persistance, and cross-domain proxy facilities through an API to the editor. The decentralized nature of mashups demands this execution engine to coordinate gadget execution and communication within the mashup. Thanks to the plugin API, extra functionality can be added to the execution engine as external modules (i.e. security, publish/subscribe, or piping modules).

End-users will find the gadgets they need from the **catalogue**. The availability of a standardized gadget and mashup description languages allows the catalogue to be decoupled from both the editor and the execution engine.

Thanks to Wirecloud, FIWare will offer technologies to build the front end layer of a new generation SOA architecture that supports the following criteria:

- End-users are fully empowered to self-serve from a wide range of available resources, providing access to content and application services, in order to set up their own personalized operating environment in a highly flexible and dynamic way ("Do it yourself", IKEA philosophy).
- Active participation of users is now enabled, they can now create resources as well as share and exchange both knowledge and resources with others and learn together through the catalogue, thus accelerating the way innovations and improvements in productivity are incorporated.

- Interaction will be adapted and relevant to context, giving the term "context" the widest possible meaning, in a way that comprises both user context (knowledge, profile, preferences, language, information about social networks the user belongs to, etc.) and delivery context (static and dynamic characteristics of the device used for access, geographical and time location, connection bandwidth, etc.). Dynamic context variability and user mobility will also be taken into consideration.

Wirecloud plans to evolve in the following services:

- Advanced Integrated Mashup Development Environment (IDE)
- Resource Catalogue and MarketPlace enabling resource (gadgets and application mashups) discovering, sharing and exploitation
- Advanced Operative Environment Management

Wirecloud base functionality can be easily enhanced by means of modules:

- Gadgets will be able to receive and publish data in a publish/subscribe fashion using the pub/sub module.
- Gadgets can use the cross-domain proxy for accessing services without taking into account whether they are accessed using cross-domain requests.
- Resources (Gadgets and Mashups) can be added to the catalogue using MDL/GDL languages.

Pending features and estimated efforts

- Distribution and federation features in the catalogue
- Integration with the Registry/Repository and marketplace.
- Integration of the MDL/GDL with the USDL.
- Support for Pub/Sub, Security and Piping in the Gadget/Plugin APIs

Technologies Used

The following technologies are used in Wirecloud:

- Server side:
  - Python
- Client side:
  - Javascript
  - HTML
  - CSS

Runtime pre-requisites

The Wirecloud platform has been built and tested on Debian Wheezy.

Known software requirements

- A Database Manager (MySQL, PostgreSQL, Sqlite3...)
- Apache
- Python 2.5, 2.6 or 2.7. Python 3 and other versions are not supported.
  - python-lxml
- Django 1.3
  - south
  - django_compressor (BeautifulSoup)
  - johnny-cache

IPR

Wirecloud Platform is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Wirecloud Platform is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details (link [1]).

- Both the Application Mashup Editor and the Mashup Execution Engine are open source products. They can be downloaded from GitHub [2].
- The Catalogue is also offered as an open source product, but its distribution and federation features are still pending. It can be downloaded as part of the Application Mashup Editor.

Publicly available documentation

Wirecloud Mashup Platform has been used and further developed in the following projects:

- The EzWeb [3] and Nuba [4] projects within the Avanza I+D program initiated by the Spanish Ministry of Industry, Tourism and Trade.
- The EC FP7 FAST [1], RESERVOIR [1], and 4CaaSt [5] projects.

The previous links contains documentation for both developers and end-users.

References

[1] https://github.com/Wirecloud/wirecloud/blob/master/LICENCE-agpl.txt
[2] https://github.com/Wirecloud/wirecloud
[3] http://ezweb.morfeo-project.org/lng/en/
[4] http://nuba.morfeo-project.org/lng/en/
[5] http://4caast.morfeo-project.eu/

Mashup Factory

Brief description

Mashup Factory is a toolset for subject matter experts or end users to compose own services and applications. The bases of the compositions are configurable telecommunications services (e.g. SMS, conferencing), Web or media services (e.g. data storage, google geocodes and maps, web dialogs) which are provided by 3rd parties. In grafical composition style own services are composed and deployed on an appropriate orchestration engine.

Programming artifacts

A graphical editor supports the combination and configuration of the services in a data flow oriented mash-up style. This results in executable service compositions which are deployed on an execution engine (supporting BPEL Business Process Execution Language). As such it does not offer any programming interfaces or APIs for developers to extend it.

Technologies Used

The following technologies are used in this asset:

- Apache Tomcat 6 web server
- eXist-db (open-source native XML database)
- mySQL database
- BPEL4WS 1.1, (Business Process Execution Language for Web Services)
- xForms (is the future of online forms as envisioned by the W3C)
- Web Service Definition Language (WSDL)
- SSL and X.509 certificates
- Linux (Ubuntu)

Runtime pre-requisites

See Technologies used.

IPR

All rights held by Deutsche Telekom Laboratorios.

Publicly available documentation

None.

LighSemantic-enabled Design Time Semi-automatic Service Composition

Brief Description

Light-Semantic-enabled Design Time Semi-automatic Service Composition is a suite for creating processes as service compositions at design time using standard modelling languages and light semantic descriptions and annotations. Service Compositions are designed using very simple composing elements taken from BPMN, such as service tasks, flows and gateways (exclusive, parallel). Processes and tasks are described using coarse-grain light semantic annotations. Not technical background is required, since the technical information is retrieved from the semantic descriptions of matching services. Service compositions are marshalled into BPMN 2.0 XML executable language for further deployment.

Programming artifacts

LighSemantic-enabled Design Time Semi-automatic Service Composition comprise the following components:

- Oryx BPMN Editor, integrated with support for light semantic based composition.
- Compel, a widget-based frontend for light semantic enable service composition, which complements Oryx
- API for exporting full executable BPMN 2.0, extending available Oryx export support.
- Light-Semantics-enable Process and Data Mediation Composer, describe in Mediation assets section
- A SOA4All POSM-based RDF repository of SWS descriptions

Technologies used

LighSemantic-enabled Design Time Semi-automatic Service Composition uses standards for service composition description and annotation. BPMN 2.0 is the language used for composition design, deployment and execution. Optionally translation into BPEL will be also supported. Semantic annotations are based on RDF/S and OWL. Service descriptions are based on SOA4All POSM, based on WSMOLite/MicroWSMO. Linkage to WSDL is based on SA-WSDL.

Runtime pre-requisites

- J2EE Web Container, such as Tomcat 5.X or superior.
- Sesame RDF repository

Known software requirements

- SOA4All SWS Description Editors, such as WSMOLite Editor (SOWER) or MicroWSMO Editor (SWEET)

IPR

All components are released under Apache 2.0 license, excepting Oryx that is released as MIT license

Publicly available documentation

- Oryx Editor Web page: [1]
- SOA4All Advanced Prototype For Service Composition and Adaptation Environment: [2]

References

[1] http://code.google.com/p/oryx-editor/
[2] http://www.soa4all.eu/file-upload.html?func=startdown&id=207

WSO2-Mediation-Layer-Extension

Brief Description

ICT-Service is a software platform based on Service oriented paradigms and standards where to expose and mediate infrastructural services and Telco Capabilities. It is a custom software developed by Telecom Italia based on a java based open source SOA stack (www.wso2.org), Apache 2.0 licenced ICT-service includes the following components: 1. A common extensible middleware OSGi based 2. A service execution layer 4. A mediation layer Particularly the mediation layer is based on the WSO2 ESB and Apache Camel framework and has the following rcapabilities: service virtualization, message mediation, security and policy, logging and monitoring, data and protocol transformations.

Programming artifacts

The ICT-Service platform is based on WSO2 OSGi based middleware that enable extendibility. In order to add a feature the developer shall basically deploy a new OSGi bundle that implements the new capability As regards mediation features, ICT-Service enables to deploy new mediation logic in three ways: - apache synapse mediator(xml file) - apache synapse custom mediator (java code) - apache camel route (java code deployed into the camel dynamic container developed by Teelcom Italia)

Technologies used

The ICT-Service platform is developed in JAVA and uses OSGi as a technology in order to add extended capabilities. Moreover, concerning mediation capability, apache Synapse mediators, apache synapse custom mediatorand apache camel routes can be added to the platform

Runtime pre-requisites

The ICT-Service platform has been built and tested on Ubuntu 10.04 LTS.

Known software requirements

JavaSE 1.6.x

IPR

ICT-Service is developed on top of WSO2 stack that is Apache 2.0 licensed. The software internally developed by Telecom Italia can be freely released too.

Publicly available documentation

- apache synapse project [1]
- wso2 soa stack [2]

References

[1] http://synapse.apache.org/
[2] http://www.wso2.com

Light-Semantics-enable Process and Data Mediation Composer

Brief Description

Light-Semantics-enable Process and Data Mediation Composer is a component developed by the SOA4All project that automates some cumbersome and prone to error process modelling (service composition) activities, based on process task light semantic annotations matched against available service and process light semantic descriptions:

- process task binding
- process task replacement (by a sub-process), fostering process template reuse
- process data flow generation

Light-Semantics-enable Process and Data Mediation Composer can be also considered of an implementation of the Service Discovery EPIC FIWARE.EPIC.Apps.Marketplace.ServiceDiscovery

Programming artifacts

Light-Semantics-enable Process and Data Mediation Composer includes a light semantics matchmaking agent compatible with SOA4All POSM annotations/descriptions implemented as RDF/S or OWL, although it also includes similar matchmaking agent compatible with WSML. Alternatively, it includes an agent supporting SOA4All Service Discovery.

Technologies used

Light-Semantics-enable Process and Data Mediation Composer relies on RDF/S, OWL, WSML and compatible reasoners, such as Sesame buit-in reasoner. Multiagent architecture of Light-Semantics-enable Process and Data Mediation Composer was implemented on Spring framework.

Runtime pre-requisites

J2EE Service Container, Axis2 Container.

Known software requirements

- SOA4All SWS Description Editors, such as WSMOLite Editor (SOWER) or MicroWSMO Editor (SWEET)

IPR

Light-Semantics-enable Process and Data Mediation Composer is released under Apache 2.0 license.

Publicly available documentation

- SOA4All Advanced Prototype For Service Composition and Adaptation Environment: [2]

SLA Model

Brief Description

The SLA Model is a comprehensive model of Service Level Agreement semantics. The model is completely technology independent: it places no constraints on what kinds of "service" are considered, nor on the technologies used to implement them. The model's purpose is to provide a precise semantics for notions such as "obligation", "warranty" and "guarantee", such that each signatory party to the SLA may understand without ambiguity what it is they are committing to when signing an SLA. This entails a precise interpretation of SLA content in terms of constraints on signatory behaviour, the violation conditions of such constraints, and the corresponding system requirements for monitoring these constraints and instigating appropriate actions in case of constraint violation.

The SLA Model will be developed as part of the FI-WARE project.

Programming artifacts

None. The SLA Model is not executable code.

Technologies used

None. The SLA Model is technology independent.

Runtime pre-requisites

None.

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory<http://en.wikipedia.org/wiki/Fair,_reasonable,_and_non-discriminatory_terms>) Terms according to pre-requisites of the FI-PPP program.

Publically available documentation

None. The SLA Model is currently under development.

Materializing Security in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Security chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how Security functions are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

Security monitoring

Baseline Assets

- IoT Internet protocols fuzzing framework: Provides the ability to enabled 6lowPan compliant device developers to test the security and robustness of their protocol stack through fuzzing.
- FI-Ware Vulnerability assessment: Offers to cloud users and managers an integrated way to assess vulnerabilities of deployed services..
- Ontology handler: Provides the ability to generate a modeled representation of IS and vulnerabilities and to work on it.
- Vulnerabilities OVAL scanner: Provides the ability to perform a deep inventory audit on installed softwares and applications, to scan and map vulnerabilities using non-intrusive techniques based on schemas Detect and to identify missed patches and hotfixes.
- NVD: a vulnerability database developed by the National Institute of Standards and Technology, provides the information about a vulnerability’s effect.
- Attack trace engine: Provides the ability conducting multihost, multistage vulnerability analysis on a network, capturing the operating system behavior and the interaction of various components in the network and identifying attack traces.
- Service-Level-SIEM: Provides an High performance and scalable SIEM.
- CVSS: Provides a universal open and standardized method for scoring IT vulnerabilities, vulnerability scores being representative of the actual risk to a FI-WARE framework.
- Visualization Framework: Enables large quantities of data to be presented to users in ways that aid their understanding of it.
- Botnet Tracking:Uses regular functions of the DNS to produce an analysis by correlating the information available

Themes

- Security monitoring

Epics

- FIWARE.Epic.Security.Security Monitoring.Normalization of heterogeneous events and correlation
- FIWARE.Epic.Security.Security Monitoring.Risk analysis
- FIWARE.Epic.Security.Security Monitoring.Decision making support
- FIWARE.Epic.Security.Security Monitoring.Visualization and reporting
- FIWARE.Epic.Security.Security Monitoring.Digital forensics for evidence

Features

- FIWARE.Feature.Security.Security Monitoring.Security compliance violation event agents
- FIWARE.Feature.Security.Security Monitoring.Sensitive services & data event agents
- FIWARE.Feature.Security.Security Monitoring.Raw sensor data & wireless event agents
- FIWARE.Feature.Security.Security Monitoring.Normalization of heterogeneous events
- FIWARE.Feature.Security.Security Monitoring.Service Level_Security Information and event management
- FIWARE.Feature.Security.Security Monitoring.Vulnerabilities impacts
- FIWARE.Feature.Security.Security Monitoring.network exploitation tool
- FIWARE.Feature.Security.Security Monitoring.Vulnerability Oval scanner
- FIWARE.Feature.Security.Security Monitoring.CVSS
- FIWARE.Feature.Security.Security Monitoring.Attack trace engine
- FIWARE.Feature.Security.Security Monitoring.Counter-measures
- FIWARE.Feature.Security.Security Monitoring.Visualisation
- FIWARE.Feature.Security.Security Monitoring.Forensics

User-Stories

- FIWARE.User-Stories.Security.Security Monitoring.Vulnerability-scanning normalization
- FIWARE.User-Stories.Security.Security Monitoring.Topological-vulnerabilities-Analysis
- FIWARE.User-Stories.Security.Security Monitoring.Visualisation4OntologyHandler
- FIWARE.User-Stories.Security.Security Monitoring.OntologyData4Visualisation
- FIWARE.User-Stories.Security.Security Monitoring.GenericData4Visualisation
- FIWARE.User-Stories.Security.Security Monitoring.IS knowledge base access

Context-based security and compliance

Baseline Assets

- Fragmento: Provides the ability to specify compliance controls as reusable process fragments.
- Compliance Governance Dashboard: Provides on-line reports on compliance.
- CRLT Compliance Request Language Tools: Provides storing and managing compliance requirements at various abstraction levels capabilities. It also provides real time compliance verification
- USDL Language: Platform-neutral language for describing services
- S&D Run-Time Framework: Framework for the automated treatment of security and dependability issues in Ambient Intelligent scenarios

Themes

- Context-based security and compliance

Epics

- FIWARE.Epic.Security.Context-based security and compliance.USDL-SEC
- FIWARE.Epic.Security.Context-based security and compliance.ServRep
- FIWARE.Epic.Security.Context-based security and compliance.Monitoring
- FIWARE.Epic.Security.Context-based security and compliance.Rules
- FIWARE.Epic.Security.Context-based security and compliance.PRRS

Features

- FIWARE.Feature.Security.Context-based security and compliance.User Request
- FIWARE.Feature.Security.Context-based security and compliance.Monitoring services
- FIWARE.Feature.Security.Context-based security and compliance.Monitoring rules
- FIWARE.Feature.Security.Context-based security and compliance.Monitoring Report
- FIWARE.Feature.Security.Context-based security and compliance.Monitoring Non compliance
- FIWARE.Feature.Security.Context-based security and compliance.Rules Manager
- FIWARE.Feature.Security.Context-based security and compliance.PRRS Rules repository
- FIWARE.Feature.Security.Context-based security and compliance.PRRS Rules modification
- FIWARE.Feature.Security.Context-based security and compliance.Security Service Deploy

User-Stories

- FIWARE.User-Stories.Security.Context-based security and compliance.USDL-SEC Service Description
- FIWARE.User-Stories.Security.Context-based security and compliance.USDL-SEC User-Context Description
- FIWARE.User-Stories.Security.Context-based security and compliance.USDL-SEC Rules Description

Identity Management Generic Enabler

Baseline Assets

- Stork: Provide a system for recognition of eID through EU member states
- Identity Management:Provide an Identity Management System both for users and devices
- White Label IdP
- Access Control: SENSEI

Themes

- Identity Management

Epics

- FIWARE.Epic.Security.IdentityEnabler.Stork
- FIWARE.Epic.Security.BulkUploadofIdentities

Features

- FIWARE.Feature.Security.TokenRequestForIoT
- FIWARE.Feature.Security.AuthorisationRequestForIoT
- FIWARE.Feature.Security.XacmlAuthorisationForIoT
- FIWARE.Feature.Security.CredentialRevocation
- FIWARE.Feature.Security.IdentityEnabler.IDM - Privacy Enabler integration

Privacy Generic Enabler

Baseline Assets

- Privacy Enabler Provides functionality to protect users privacy
- Idemix - Privacy-Preserving Credential Library
- Idemix - Credential-based Authentication Engine
- Accountable privacy policies

Themes

- Privacy Management

Epics

- FIWARE.Epic.Security.IdentityEnabler.IDM-IDEMIX_integration
- FIWARE.Epic.Security.Interworking_between_IDM_and_PPL
- FIWARE.Epic.Security.Accountability

Data Handling Generic Enabler

Baseline Assets

- PrimeLife Policy Engine: PPL

Themes

- Data Handling

Optional Security Enabler

Baseline Assets

- Database Anonymization Optional Asset
- Secure Storage Service (SSS) Optional Asset (Thales)
- Morphus Optional Asset (INRIA)

Themes

- Malware Detection
- Database anonymization
- Secure storage

Epics

- FIWARE.EPIC.Security.OptionalSecurityEnablers.Morphus
- FIWARE.EPIC.Security.OptionalSecurityEnablers.DBAnonymization
- FIWARE.EPIC.Security.OptionalSecurityEnablers.SSS

Features

- FIWARE.Feature.Security.OptionalSecurityEnablers.Usdlsec4ContentBasedSecurity

User-Stories

- FIWARE.Story.Security.OptionalSecurityEnablers.ServiceSerialization

Topics still being addressed at high-level

Epics

- FIWARE.Epic.Security.Security Monitoring.CEP
- FIWARE.Epic.Security.Security Monitoring.CMDB

IoT Internet protocols fuzzing framework

Brief Description

The fuzzing asset is a framework designed to enabled 6lowPan compliant device developers to test the security and robustness of their protocol stack through fuzzing.

The fuzzer builds on Scapy's features. It incorporates features for replicate common interactions between devices, called Scenarios, represented in a simple XML Language. The fuzzer also supports the application of different algorithms over created scenarios. These fuzzing algorithms can be either predefined (delivered as a library, or implemented directly by the framework user.

IPR

This framework is licensed under the GPLv2 license

Runtime pre-requisites

The fuzzer platform has been built and tested on Ubuntu 11.04.

Known software requirements

- Scapy [1]
- Python 2.6 or higher [2]
- Mercurial (for the installation of the fuzzer only [3]

Publicly available documentation

-

References

[1] http://www.secdev.org/projects/scapy/
[2] http://http://www.python.org/
[3] http://mercurial.selenic.com/

FI-Ware Vulnerability assessment

Brief Description

OVALIZER aims at offering cloud users and managers an integrated way to assess vulnerabilities of deployed services. This service detects the presence of known configuration vulnerabilities by comparing the characteristics of the assessed service to a set of vulnerability definitions extracted from an OVAL (Open Vulnerability and Assessment Language) repository.

OVALIZER is composed of three main components :

- An OVAL repository
- An OVAL specifications processing engine mapping oval to cfengine policies
- An enhanced cfengine agent on any managed system

IPR

The final licensing scheme selection for this asset has not yet been finalized

Ontology handler

Brief description

The ontology handler serves as a knowledge base to store information relative to IS infrastructure and their associated security data using the Security Ontology. Each ontology instance corresponds to a version of a system and the vulnerabilities, threats, and safeguards that are related to its assets. These security data are retrieved from the Vulnerability DB. Finally, the ontology repository embeds mechanisms to create, save and retrieve instances of the Security Ontology and mechanisms to add, delete, and rename individuals and to set links between the individuals with predefined properties. These functionalities are exposed in an extern interface in order for other components to remotely manipulate ontologies.

Programming artifacts

Main entities are:

- Ontology: this class represents a specific instance of the security ontology. Its content can be manipulated on memory and persistence is maintained by storing it on hard drive.

- OntologyRepository: this entity represents the database containing all the instances of the Security Ontology. Ontology instances can only be created and accessed through this class.

- OntologyRepositoryHandlerInterface: this class is the interface containing all the services provided by the component.

- OntologyRepositoryHandlerImpl: this class is the implementation of the provided services.

- OntologyRepositoryServer: this class is responsible for initializing the repository and exposing its services so they can be remotely accessible.

- VulnerabilityCollecte: this class is responsible for accessing the Vulnerability DB, interrogating it with assets & CPEs, retrieving the corresponding XML file and adding the contained security data to the ontology instance. It is also in charge of parsing Vulnerability notifications and updating the ontologies with the provided data.

Technologies Used

The ontology repository has been implemented in Java. Jena and Jess libraries were used to manipulate ontologies. Its APIs were defined in accordance with the principles of the OSS/J specification. They are available remotely by means of a Web Service.

Runtime pre-requisites

Apache CXF

IPR

To be completed.

Publicly available documentation

To be completed.

Vulnerabilities OVAL scanner

Brief description

Nessus is designed to automate the testing and discovery of known security problems, allowing to correct problems before they are exploited. Nessus will be compatible OVAL Through a converter.

Open Vulnerability and Assessment Language (OVAL™) is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. OVAL™ includes a language used to encode system details, and an assortment of content repositories held throughout the community.

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.); and reporting the results of this assessment. Source : oval.mitre.org.

Programming artefacts

Nessus uses a client server technology. Servers can be placed at various strategic points on a network allowing tests to be conducted from various points of view. A central client or multiple distributed clients can control all the servers.

Nessus checks database is updated on a daily basis and can be retrieved with the command nessus-update-plugins. Nessus has the ability to detect the remote flaws of the hosts on the network, but their local flaws and missing patches as well.Nessus has the ability to test SSLized services such as https, smtps, imaps, and more.

Nessus has been built so that it can easily scale. Nessus Security Scanner includes NASL, (Nessus Attack Scripting Language) a language designed to write security test easily and quickly. Nessus gives you the choice between performing a regular non-destructive security audit on a routinely basis, or to throw everything you can at a remote host to see how will it withstands attacks from intruders.

The language OVAL 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.) and Reporting the results of this assessment.

Technologies Used

Open Vulnerability and Assessment Language (OVAL™)

Runtime pre-requisites

Windows and Linux environments

IPR

Nessus is public domain software released under the GPL

Publicly available documentation

http://www.oval.mitre.org

NVD

Brief Description

NVD (National Vulnera bility Database)is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance.

NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. Fortunately, the NVD classifies the effect of a vulnerability in two dimensions: exploitable range and consequences.

• exploitable range: local, remote

• consequence: confidentiality loss, integrity loss,denial of service, and privilege escalation

Programming artefacts

NVD contains content (and pointers to tools) for performing configuration checking of systems implementing the FDCC(Federal Desktop Core Configuration) using the Security Content Automation Protocol (SCAP).

A SCAP vulnerability database is a product that contains a catalog of security related software flaw issues labeled with CVEs where applicable. This data is made accessible to users through a search capability or data feed and contains descriptions of software flaws, references to additional information (e.g., links to patches or vulnerability advisories), and impact scores..

Technologies Used

SCAP Security Content Automation Protocol version 1.0, Specification NIST SP800-126

Runtime pre-requisites

Vulnerability Search Engine (CVE software flaws and CCE misconfigurations)

SCAP Data Feeds (CVE, CCE, CPE, CVSS, XCCDF, OVAL)

IPR

NVD is a product of the NIST Computer Security Division and is sponsored by the Department of Homeland Security’s National Cyber Security Division. It supports the U.S. government multi-agency (OSD, DHS, NSA, DISA, and NIST) Information Security Automation Program. It is the U.S. government content repository for the Security Content Automation Protocol (SCAP).

publicly available

NVD: http://nvd.nist.gov/

FDCC: http://nvd.nist.gov/fdcc/index.cfm

FDCC Checklist: http://web.nvd.nist.gov/view/ncp/repository

Attack trace engine

Brief description

The objective is to preemptively identify the attack trace and to avoid the access to the assets being the target of an attack. An attack can impact a service by compromising resources that are needed to deliver this service. Failures to complete a task could disturge the service delivery effectiveness or cause the service to fail. In a FI-WARE, it will possible to assign a value to sensitive services for example and associeted resources, that provides a basis for measuring service impact of an attack. The attack trace engine is used to understand vulnerabilities an attacker can exploit next, and based on the current state of the services, anticipate the impact of attack step, then to suggest the best actions to minimize future impact. The discovery of attack traces requires a reasoning engine analyzing the inputs from vulnerability Oval scanners,network topology data base and Common Vulnerability Scoring System.

Programming artefacts

Attack trace engine models the interaction of component vulnerabilties with system and network configurations.

The information in the vulnerability database provided by the bug-reporting community, the configuration information of each machine and the network, and other relevant information are all encoded as Datalog facts.

The reasoning engine consists of a collection of Datalog rules that captures the operating system behavior and the interaction of various components in the network. Thus integrating information from the bug-reporting community and off-the-shelf scanning tools in the reasoningmodel is straightforward.

Technologies Used

- Datalog Language (Deductive Database Programming), a subset of Prolog, modeling language for the elements in the analysis (vulnerability specification, configuration description, reasoning rules, operating-system permission and privilege model, etc..

- Reasoning rules specifying semantics of different kinds of exploits, compromise propagation, and multihop network access.

- Host Access Control List specifying all accesses between hosts that are allowed by the network.

- Policy specification defining user access rules to information and services.

- Binding information mapping a data symbol to a path on a machine.

Runtime pre-requisites

Network topology, Vulnerabilities collection from Oval scanner, ITAC data base and CVSS scoring.

Otherwise, for impact analysis, we assume that service workflows are known, and tells what resources are needed to the service delivery.

IPR

Usage submited to preliminary autorisation of the University of Kansas

Publicly available information

http://people.cis.ksu.edu/~xou/mulval/

Service-Level-SIEM

Brief Description

One of the main constraints of commercial and non-commercial SIEM products is the scalability of current solutions. Current solutions depend largely on centralised rule processing that forces that single nodes process the full event traffic bounding the capacity of the system by the capacity of a single node.

The idea is to run a high-performance and scalable event correlator engine on top of an existing open source SIEM (such as OSSIM), extending in this way the correlation capabilities of OSSIM.

The number of events having to be handled by SIEMs in real large deployments is duplicating every year. This exponential increase in the volume of alerts is rendering obsolete underlying correlation technology of SIEMs due to its centralised nature. The next generation of SIEMs should be able to correlate large volumes of information in a distributed manner.

In addition to the increase in the number of security events that can be processed per second, the new correlation engine will provide also elasticity for the event processing to minimize the used resources to the minimal required for the incoming event load. Currently, over-provisioning is being used resulting in very high costs.

Programming artifacts

- Java
- C++

Technologies used

OSSIM Open Source SIEM.

Runtime pre-requisites

OSSIM must be instaled

IPR

OSSIM is a third party product licensed under GPL.

The Event Correlator engine developed by Atos will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program

Publicly available documentation

- http://www.ossim.net
- http://www.massif-project.eu

CVSS

Brief description

The Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. CVSS consists of 3 groups: Base, Temporal and Environmental. Each group produces a numeric score ranging from 0 to 10, and a Vector, a compressed textual representation that reflects the values used to derive the score. The Base group represents the intrinsic qualities of a vulnerability. The Temporal group reflects the characteristics of a vulnerability that change over time. The Environmental group represents the characteristics of a vulnerability that are unique to any user's environment. CVSS enables IT managers, vulnerability bulletin providers, security vendors, application vendors and researchers to all benefit by adopting this common language of scoring IT vulnerabilities.

CVSS quantitative model ensures repeatable accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores. Thus, CVSS is well suited as a standard measurement system for industries, organizations, and governments that need accurate and consistent vulnerability impact scores.

Two common uses of CVSS are prioritization of vulnerability remediation activities and in calculating the severity of vulnerabilities discovered on one's systems. The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities.

CVSS quantitative model ensures repeatable accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores.

Programming artefacts

National Vulnerability Database (NVD) supports the Common Vulnerability Scoring System (CVSS) version 2 standard for all CVE vulnerabilities. NVD provides CVSS 'base scores' which represent the innate characteristics of each vulnerability. NVD does provide a CVSS score calculator to allow you to add temporal data and to even calculate environmental scores (scores customized to reflect the impact of the vulnerability on your organization).

NVD provides severity rankings of "Low," "Medium," and "High" in addition to the numeric CVSS scores but these qualitative rankings are simply mapped from the numeric CVSS scores:

1. Vulnerabilities are labeled "Low" severity if they have a CVSS base score of 0.0-3.9.
2. Vulnerabilities will be labeled "Medium" severity if they have a base CVSS score of 4.0-6.9.
3. Vulnerabilities will be labeled "High" severity if they have a CVSS base score of 7.0-10.0.

Technologies Used

The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental.

Runtime pre-requisites

Common Vulnerability Scoring System Version 2 Calculator. Score.

IPR

CVSS is a product of the NIST Computer Security Division and is sponsored by the Department of Homeland Security’s National Cyber Security Division. It supports the U.S. government multi-agency (OSD, DHS, NSA, DISA, and NIST) Information Security Automation Program.

Publicly available

CVSS: http://nvd.nist.gov/cvss.cfm?version=2

Common Vulnerability Scoring System Calculator: http://nvd.nist.gov/cvss.cfm?calculator&version=2

CVSS Standard specification: http://www.first.org/cvss/cvss-guide.html

CVSS v2 impact vector specification: http://nvd.nist.gov/cvss.cfm?vectorinfo&version=2.

Visualization Framework

Brief Description

The Visualisation Framework offers a visualisation service that allows users to visualise data from multiple network components, e.g. IDS, network probes. The user accesses the visualisation service through a standard web-browser connected to the web-application server using some network connection (such as the Internet). The user will experience a single integrated application showing multiple visualisations. Behind the scenes, the browser will compliment the information from the visualisation server with data and functionality directly from the Internet.

The visualisation service is aimed at users who want to either:

- Monitor the health and performance of their network
- Investigate a suspected problem
- Investigate an automatically generated alarm
- Understand the actions of and therefore gain confidence in automatic systems such as IDS, reaction and remediation components

For all of these actions the user will follow a similar pattern of creating, interacting with, modifying and eventually removing visualisations.

The Visualisation Framework enables the data from new components to be visualised. The component must export data in one of the supported data formats to either the framework's messaging queue or to a database that is accessed by the visualisation server.

Programming artifacts

Methods offered to end-users

- Add new visualisation
- Modify visualisation
- Remove visualisation

Methods offered to administrators

- Add new data source (supported data format)

Adding a new data source that does not use the supported data formats is not currently offered as a service. Adding new visualisations that do not use the supported data formats is also not currently offered as a service. Both of these require updates to the software, as the data formats are hard-coded.

Technologies Used

The following technologies are used in the repository asset

- Java EE
- Spring Framework
- Apache Tomcat. The web applications are deployed in a Tomcat container.
- Apache ActiveMQ
- Apache Derby
- Apache Camel
- JAXB
- Hibernate

The following technologies are used to implement the visualisations:

- Google Charts

- Google Visualization
- SIMILE
- Script.aculo.us
- Prototype.js

Runtime Pre-requisites

The Visualisation Framework is specified to retrieve data from Java Messaging Service (JMS) queues.

IPR

There are no patents relating to this asset. However, it is the property of Thales UK. The product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) terms according to the pre-requisites of the FI-PPP program.

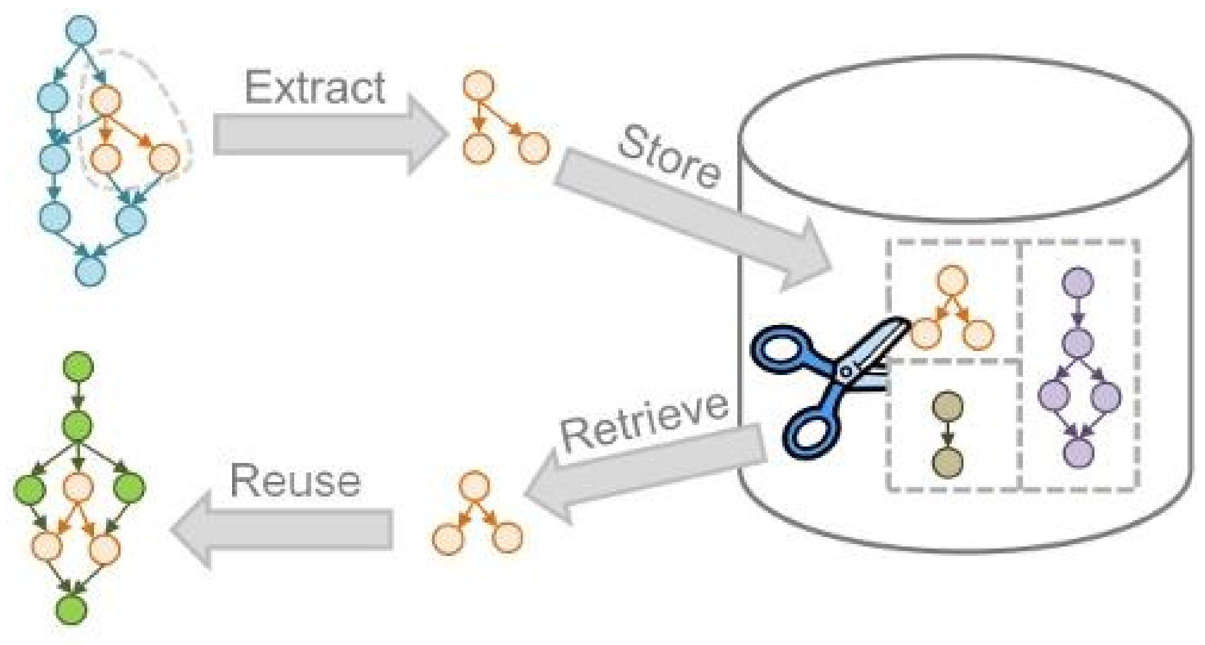
Publicly available documentation

- FP7 INTERSECTION project (http://www.intersection-project.eu/).The main reference report for this asset is D3.2 Final Security Framework Architecture and Tool Interaction Specification.

Fragmento

Brief Description

Companies increasingly have to pay attention to compliance. Flexibly reacting to requirements coming from laws, regulations, and internal policies becomes an essential part of business process management. Compliance refers to the measures that need to be taken in order to adhere to these requirements. In order to comply, companies need to perform profound changes in their organizational structure, processes and their supporting infrastructure. At this, process-awareness is a basic prerequisite for ascertaining to operate in a compliant way



Compliance controls as reusable process fragment

The Fragmento repository focuses on the application of the emerging concept of process fragments in the field of compliance management in process-based applications. This tool proposes to specify compliance controls as reusable process fragments. We understand a process fragment as a connected, possibly incomplete process graph which may also contain additional artifacts like the fragment context. A process fragment is not necessarily directly executable as some parts may be explicitly stated as opaque in order to mark points of variability

Programming artifacts

A repository for process artifacts needs to account for persistence, storage, retrieval and version management of all artifacts related to a process.

We can distinguish different types of artifacts that are related to a process or to a process fragment in a service-oriented environment.

• A process or process fragment model, either in standard BPEL or in an extended or modified version of BPEL

• An according WSDL document (for specifying the interface)

• A deployment descriptor according to the process or the process fragment

• Additional information for a process modeling tool for instance to store X and Y coordinates of activities

• A view transformation rule for process view transformations

• Annotation documents (e.g., a policy specified in WS-SecurityPolicy)

• The repository assigns uniquehttps://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/skins/common/images/button_hr.png identifier to each artifact being stored. The usage of unique identifiers allows creating arbitrary relations between artifacts. The above listed types of artifacts are typically serializable to XML, therefore the repository mainly needs to store XML artifacts. Additional metadata, such as a description, keywords and a name are also applicable to all of the listed types.

Technologies used

- XMLO repository

- SQL database (we have made good experience with PostGreSQL)

- Apache Tomcat V5.5

- Apache Axis 2 V1.4

- JDK 1.5 and according IDE for source code modifications

- For testing of the the Fragmento Web service we recommend SoapUI

This tool is used with BPEL technology. It means that the input and output must be under BPEL format

Runtime pre-requisites

Fragmento provides all required functionality via Web service interfaces which can be exploited for integration with rich client applications, e.g., based on Eclipse.

IPR

Fragmento is free for reuse, the license is Apache 2 Open Source.

Publicly available documentation

http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/source.htm

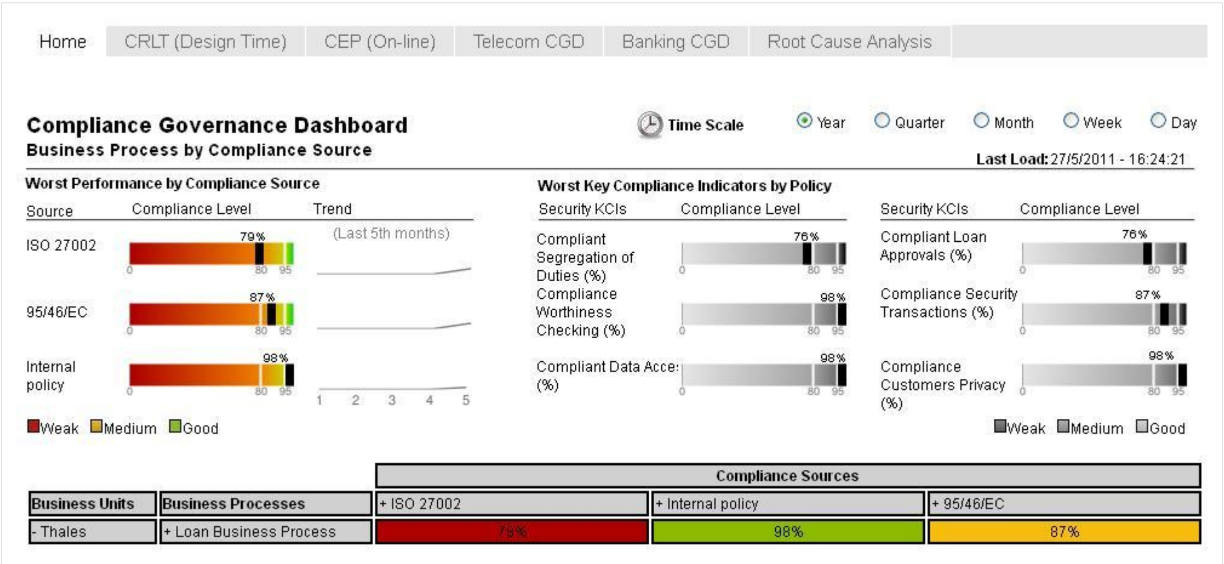
Process Illustrator: http:/ / www. youtube. com/ user/ WSO2TechFlicks#p/ a/ 733061E8B904B0AE/ 0/ rG08x4dMWzI

Source code: http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/schumm/indexE.php#downloads

Compliance Governance Dashboard

Brief Description

Compliance Governance Dashboard (CGD) aims at reporting on compliance, creating an awareness of possible problems or violations, and facilitating the identification of root-causes for noncompliant situations. For that, CGDs concentrates on the most important information at a glance, condensed into just one page.



Compliance Governance Dashboard
Business Process by Compliance Source

[✓ Year] Quarter Month Week Day

Last Load: 27/5/2011 - 16:24:21

Worst Performance by Compliance Source

Source	Compliance Level	Trend (Last 5th months)
ISO 27002	79%	
95/46/EC	87%	
Internal policy	98%	

🟥Weak 🟨Medium 🟩Good

Worst Key Compliance Indicators by Policy

Security KCIs	Compliance Level	Security KCIs	Compliance Level
Compliant Segregation of Duties (%)	76%	Compliant Loan Approvals (%)	76%
Compliance Worthiness Checking (%)	98%	Compliance Security Transactions (%)	87%
Compliant Data Acce: (%)	98%	Compliance Customers Privacy (%)	98%

■Weak ▨Medium □Good

		Compliance Sources		
Business Units	Business Processes	+ ISO 27002	+ Internal policy	+ 95/46/EC
- Thales	+ Loan Business Process	79%	98%	87%

Governance Dashboards concentrates on the most important information at a glance, condensed into just one page

Programming artifacts

Warehousing business process data is the first step toward enabling the analysis of business performance and of problems in business execution. Although many solutions for data warehousing exist, their focus is typically the core business of a company, and compliance is usually not taken into account.

Dashbord provides a conceptual model for compliance dashboards that covers a broad class of compliance issues. Identify the key abstractions and their relationships.

It Defines, besides the conceptual abstractions, a user interaction and navigation model that captures the way the different kinds of users need to interact with the dashboard, to minimize the time to accesses spent in getting the information users need and to make sure that key problems do not remain unnoticed.

It derives a model that is in line with the criteria and approach that auditors have to verify compliance.

Technologies used

Eclipse is used to develop this asset with this technologies: • JSP for the User Interface • Tomcat Apache • MySQL

ETL's are used to extract and to load the events data to data base.

Runtime pre-requisites

As ETL's are used to manage data; data entries must be formatted in order to be exploited by the Compliance Governance Dashboard

IPR

Code developed and deployed just using open source technologies, moreover everything is freely available on the web.

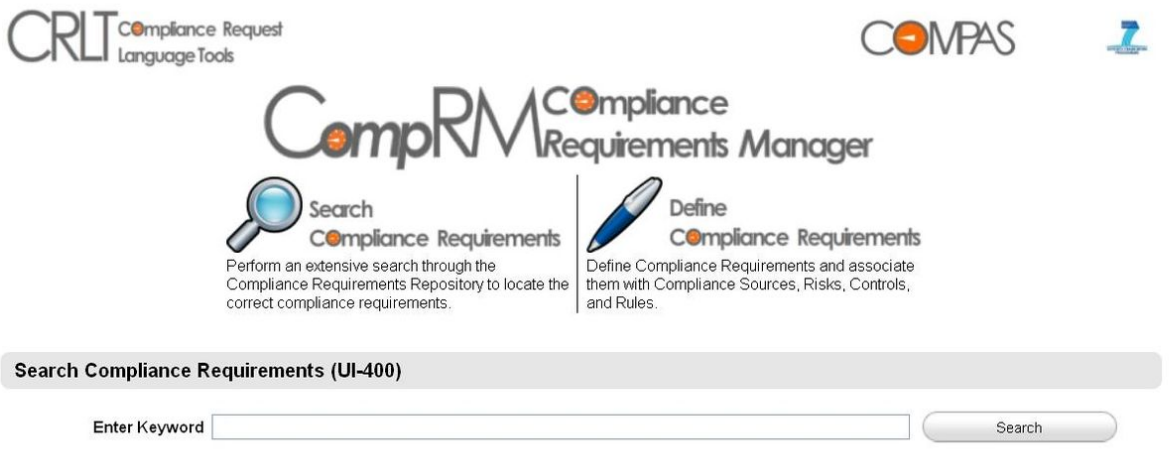
Publicly available documentation

Information available at: http://compas.disi.unitn.it:8080/CGDs/main.jsp

CRLT Compliance Request Language Tools

Brief Description

In today's business environment, organizations are required to cope with increasing number of compliance regulations, such as Sarbanes-Oxley, Basel II, IFRS, HIPAA, etc., that oblige them to review their business processes and ensure that these processes are compliant to the requirements set forth. In a broader perspective, compliance is ensuring adherence to a set of prescribed and/or agreed norms. These norms originate from various sources, including laws and regulations, public and internal policies, standards, customer preferences, partner agreements and jurisdictional provisions. In order to manage compliance throughout all phases of business process lifecycle, it must be possible to define and subsequently integrate compliance specifics into business processes and enterprise applications, and assure compliance starting from the process analysis and design phase. Compliance Request Language Tools (CRLT) serves this purpose. CRLT allows storing and managing compliance requirements and relevant specifics at various abstraction levels in a Compliance Repository, and check end-to-end business processes for compliance against the set of applicable constraints during design-time. CRLT (currently) has three main components.


CRLT Compliance Request Language Tools
COMPAS
CompRM Compliance Requirements Manager
Search Compliance Requirements
Define Compliance Requirements
Search

Perform an extensive search through the Compliance Requirements Repository to locate the correct compliance requirements.

Define Compliance Requirements and associate them with Compliance Sources, Risks, Controls, and Rules.

Search Compliance Requirements (UI-400)

Enter Keyword

Storing and managing compliance requirements

Programming artifacts

CRLT has three main components:

- Compliance Requirements Manager (CompRM) is a web-based application that allows compliance and business experts to define, store and manage compliance requirements and relevant concepts (sources, risks, etc.) in the CRR.

- Rule Modeler is a standalone component of the CRLT that enables compliance experts to visually create pattern-based representations of controls and transform these expressions into compliance rules (as formally represented logical formulas).

- Design-time Compliance Request Manager (DCRM) is a web-based application for formulating compliance requests at design-time phase of business process compliance for verifying compliance targets (mainly business process specifications) against relevant compliance requirements.

Technologies used

• JSP

• MySQL

• DSL Tool of Microsoft.

Runtime pre-requisites

Software requirement:

- Windows Vista or Windows 7
- Microsoft Visual Studio 2008 (professional, premium or ultimate editions)
- Microsoft Visual Studio 2008 SDK (Software Development Kit)

IPR

Open source

Publicly available documentation

Information available at: http://eriss.uvt.nl/compas/

USDL Language

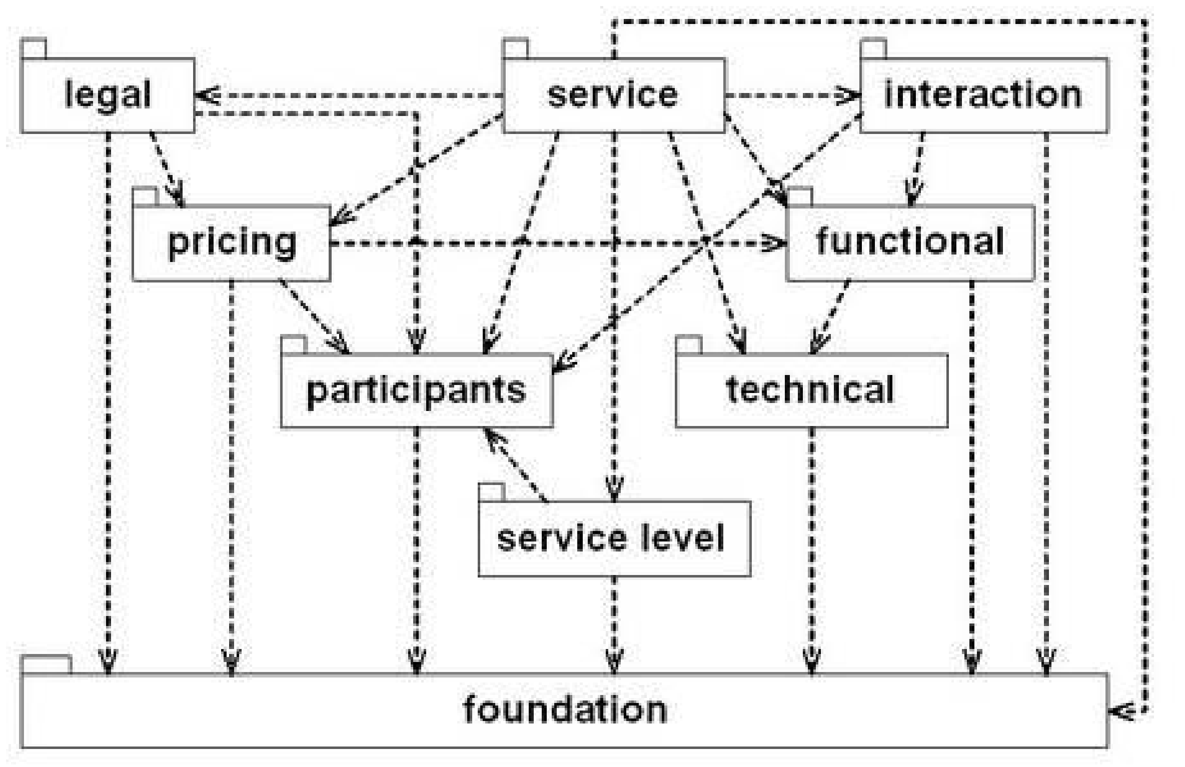
Brief Description

The Unified Service Description Language (USDL) is a platform-neutral language for describing services. It was consolidated from SAP Research projects (FAST, RESERVOIR, MASTER, ServFace, SHAPE, SLA@SOI, SOA4ALL) concerning services-related research as an enabler for wide leverage of services on the Internet. SAP Research believes that 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.

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) and infrastructure (e.g. CPU and storage services).

USDL Specifications

USDL on a whole is made up of a set of modules, each addressing different aspects of the overall service description. Modularization was introduced to improve readability of the model, which drastically grew in size compared to its predecessor. The modules have dependencies among each other (shown in the figure below), as they may reuse concepts from other modules. Currently, there are 8 modules in the set that constitutes USDL version 3.0. All of them are covered in the milestone 5 release of USDL.



USDL Module structure

USDL-SEC in FI-WARE

The security description in USDL is not yet specified. Some first attempts started internally SAP Research without coming out with concrete results. The main task in WP8 will be the extraction of the main security functionalities proposed in the enablers in order to map them to a more generic security description. USDL-SEC may refer to existing security standards like WS-SecurityPolicy. Security is modeled in USDL on an abstract level: Security Requirement Levels act as profile mechanism. (The mapping to the concrete technical artefacts: WS-securityPolicy/XACML might be supported by a security ontology and a platform dependent instantiation of this security ontology).

The Second task will be the provisioning of a technical solution to map and transform the security specifications into security functional services.

Programming artifacts

• No programming artifacts for the specification

Technologies used

• USDL Editor: The USDL3 Editor is a graphical user interface to create descriptions of your services and save them into USDL files. The files can then be deployed on marketplaces and traded accordingly. Two different versions of the USDL3 Editor are available, a web-based editor and a stand-alone version

Runtime pre-requisites

• N/A

IPR

• http://www.sap.com/corporate-en/our-company/legal/copyright/index.epx

Publicly available documentation

Internet of services: http://www.internet-of-services.com/index.php?id=260&no_cache=1&L=0

S&D Run-Time Framework

Brief Description

The objective of this asset is to provide a framework for the automated treatment of security and dependability issues in Ambient Intelligent scenarios

The S&D Run-Time Framework is implemented as a service running in a device, on top of the operating system, and listening to applications requests. Applications send requests in order to fulfill their security requirements. These requests, once processed by the S&D Run-Time Framework, are translated into S&D Solutions that may be used by the application. As a result of the selection process, the SRF instantiates S&D Patterns or Integration Schemes by means of Executable Components that are accessible by applications.

The main components of its architecture are:

• S&D Manager: Core component of the system which controls the rest of the components of the system. Its main functionalities are:

```
 - Receives the S&D request from applications to return back an Executable Component that fulfils the S&D request.
 - Responsible for the activation and deactivation of the Executable-Components.
 - Takes the necessary actions when it is informed by the Monitoring Service when a rule violation occurs.
 - Acts as a middleware with the console receiving the requests from it and returning back the results
 - Sends the monitoring rules to the Monitoring Service to be monitored.
```

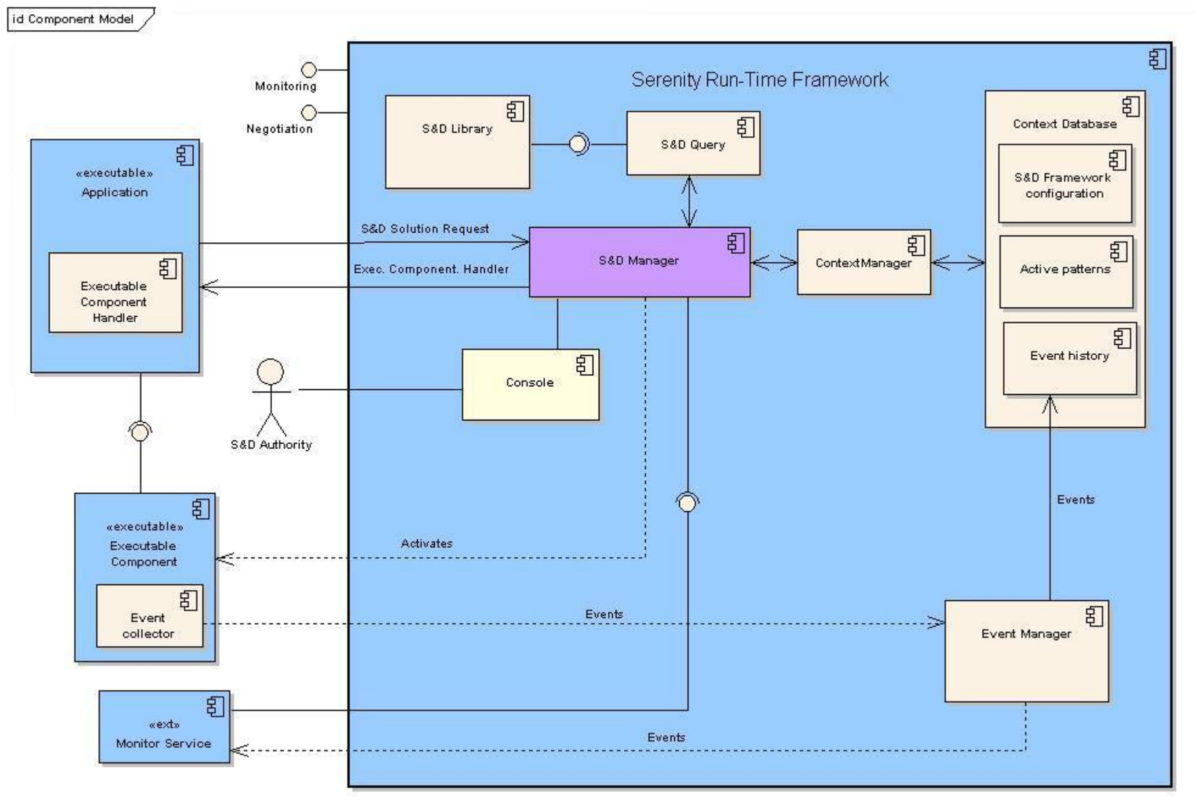
• S&D Library: Local S&DSolution repository which stores the S&DClasses, Patterns and Implementations which are specific to the platform and the applications that may be used in the device.

• S&D Query: Responsible for consulting the S&D Library database and retrieving the S&DClasses, S&DPatterns or S&DImplementations that fulfil the request.

• Context Manager: Records the data related to the context (S&D Framework configuration, active patterns, event history) to be used by the S&D Manager to select the most appropriate S&DSolution for the given scenario.

• Event Manager: In charge of receiving the events from the Event Collectors and sending them both to the Monitoring Service in order to be analyzed and to the Context Manager to in order be stored.

• Console: Main interface through which the S&D Manager will contact the SRF administrator and vice versa.



S&D Framework main components

Programming artifacts

Java API provided

Technologies used

WebServices: Java – RMI

Runtime pre-requisites

Applications should implement the Java interface defined by the Run Time Framework

IPR

The S&D Run-Time Framework tool has been developed by Atos in Serenity project and will be integrated in FI-WARE under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program

Publicly available documentation

Information available at: http://www.serenity-project.org/-Activities-.html

Stork

Brief Description

STORK implements an EU wide interoperable system for recognition of eID and authentication that enables businesses, citizens and government employees to use their national electronic identities in any Member State. It helps to simplify administrative formalities by providing secure online access to public services across EU borders.

The STORK interoperable solution for electronic identity (eID) is based on a distributed architecture that paves the way towards full integration of EU e-services while taking into account specifications and infrastructures currently existing in EU Member States.

The solution enabling secure and mutual recognition of national electronic identities between participating countries comprises:

• Detailed technical and functional specifications / design for the common architectural framework components and their interfaces,

• The implementation of two eID models by the Member States according to the common specifications, a pan-European quality authentication assurance scheme establishing the mapping of national authentication level onto STORK QAA levels to ensure cross-border and cross-service interoperability of eID,

• An analysis of legal issues identified for pan-European authentication, eID process flows for authentication, attribute transfer and e-Signature transfer

• Multi-dimensional results from the application and uptake by citizens, eGovernment Service Providers and Identity/Attribute Providers of all these sub-assets in real life cross-border services in a wide array of pilots covering from authentication to eGovernment portals to Saferchat for underage students, university student mobility, electronic delivery or change of address between Member States

• A multi-faceted study of the long-term sustainability issues around both the eID Interoperability Platform and piloted services beyond the duration of STORK.

STORK has been able to integrate many heterogeneous eID solutions implemented by the Member States into a common interoperability framework without interfering with the existing infrastructure, connecting all the various authentication methods with transparency, in such a way that any of these methods will allow users to present their certified personal data to foreign administrations.

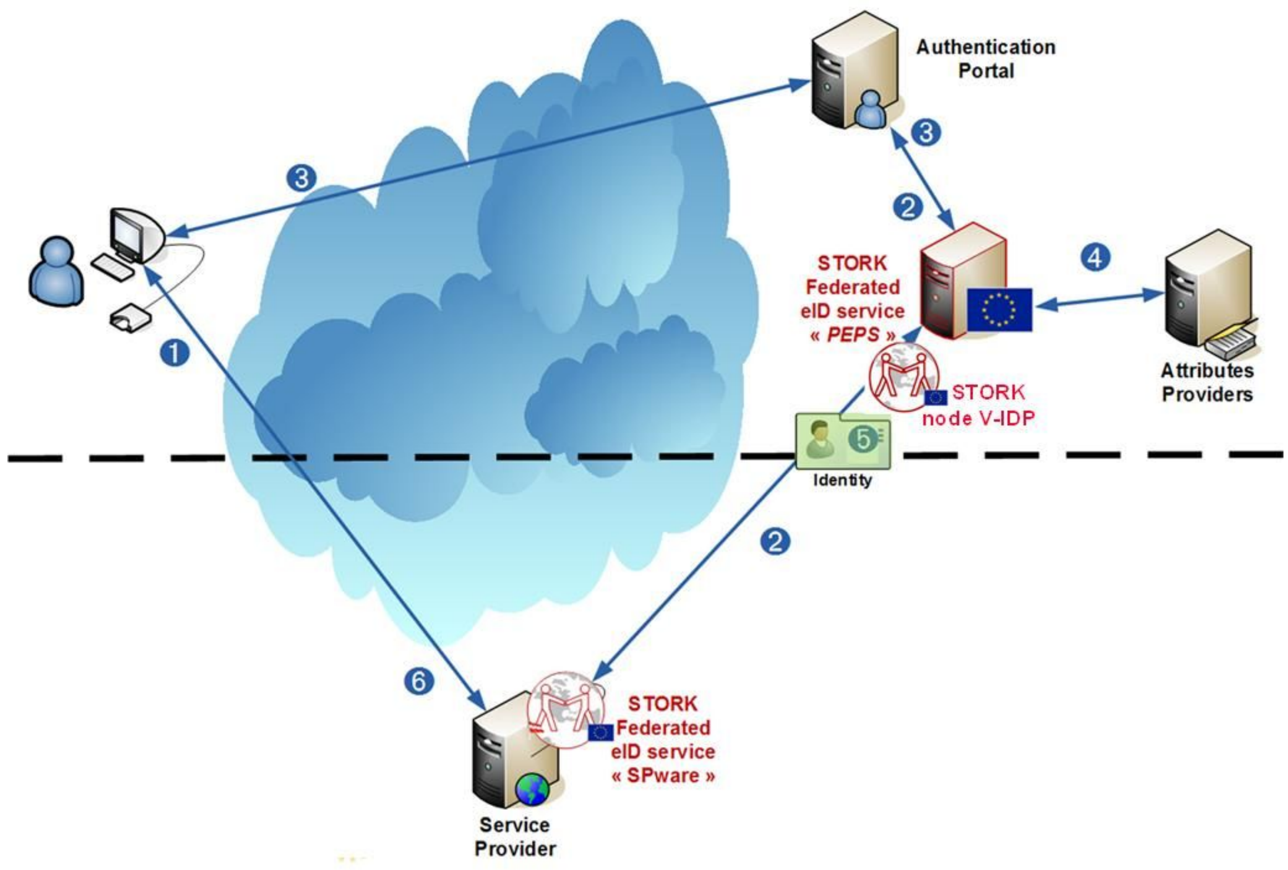
This has to respect the administrative culture and choices of Member States as diverse as to issuing eID on national, regional, or local level; technological choices such as smartcards, mobile ID, or username passwords; or centralized authentication providers vs. distributed approaches. Despite the authentication process followed, there is no storage in PEPS; only information transfer from one form to another. Thus, any information provided, remains private.

It consists of interoperable distributed platforms for eID that exchange identity data based on SAML 2.0 OASIS specification. Each member state deploys one or several of these platforms. The basic principle behind these platforms is to respect whatever national eID infrastructure is in place in each member state. The platforms combine a centralized and a distributed approach when it comes to deploy and install interoperable eID platforms in each member state.

The common framework includes two deployment options, as “PEPS (for Pan-European Proxy Service)” for a centralized implementation at Member State level, and as “MW (for middleware)” for a distributed approach. These implementation options reflect choices of Member States whether to delegate authentication decisions to centralized components or whether to keep those under service providers’ responsibility.

While the centralized and distributed deployment options reach deep into infrastructure decisions of member states, the common framework has been designed so that seamless cross-overs are provided using SAML, ensuring in any case that, at a pan-European level, the overall architecture benefits from the advantages of a distributed topology with clean and clear responsibilities of the participating entities.

The role of the STORK platform is to identify a user who is in a session with a service provider, and to send his data to this service. Whilst the service provider may request various data items, the user always controls the data to be sent. The explicit consent of the owner of the data, the user, is always required before his data can be sent to the service provider. The STORK platform does not store any personal data, so no data can be lost. This user-centric approach is in line with the legislative requirements of all the various countries involved that oblige concrete measures be taken to guarantee respect of a citizen's fundamental rights, such as his privacy.



Stork Overview

Programming artifacts

Integration package for services providers with:

• Technical documentation

• Configuration documentation

• Set up documentation

• Required software libraries

Technologies used

The core of the platform is built with Java, and the interchanged messages are standard OASIS SAML 2.0 messages as it seeks to be technologically neutral and based on open standards. Several well established libraries are used such as OpenSAML, Bouncy Castle Cryptography, Log4J, JUnit, Joda Time, Struts and Spring

Runtime pre-requisites

Integration packages for service providers will be available for:

- Java
- PHP
- .net

IPR

Main and core components of Stork will be under EUPL 1.1 license.

Some ancillary components will be under another Open Source license to be established.

Publicly available documentation

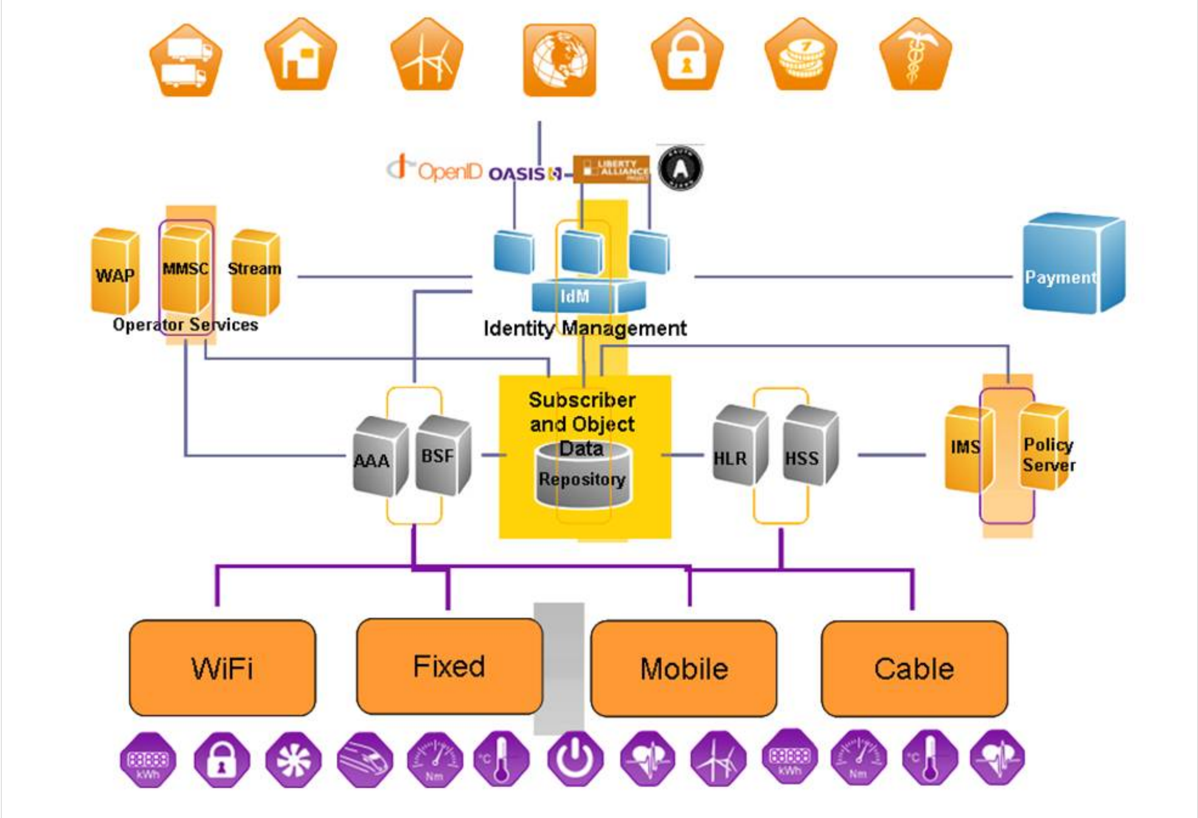
Information available at: https:/ / eid-stork. eu/ index. php?option=com_processes& act=list_documents& s=1& Itemid=60&id=312

Identity Management

Brief Description

The Identity Enabler acts as the identity provider; it provides federated identity for Web-SSO and attributes for an authenticated user. Links the network

access authentication with the application level access authentication.



IDM Overview

Programming artifacts

Integration package for services providers with:

- Functional description
- Interface description

Technologies used

The Identity Management System is JAVA based and using SAML2.0.

Runtime pre-requisites

No specific requirements

IPR

The basic functionality of identity management enabler is protected by IPRs.

Publicly available documentation

None

White Label IdP

Brief Description of the Concept

The Cloud User Management is a Lab API offering of Deutsche Telekom Developer Garden [1], which enables developers and suppliers of Web applications to store user data and relevant profile information safely and reliably in the cloud. The User Management only needs to be integrated into the Web application and then can be used from the cloud at any time, as required and free of charge. The Cloud User Management provides the typical basic functions of user management:

• Registration of user accounts

• Login for registered users by means of user name and password

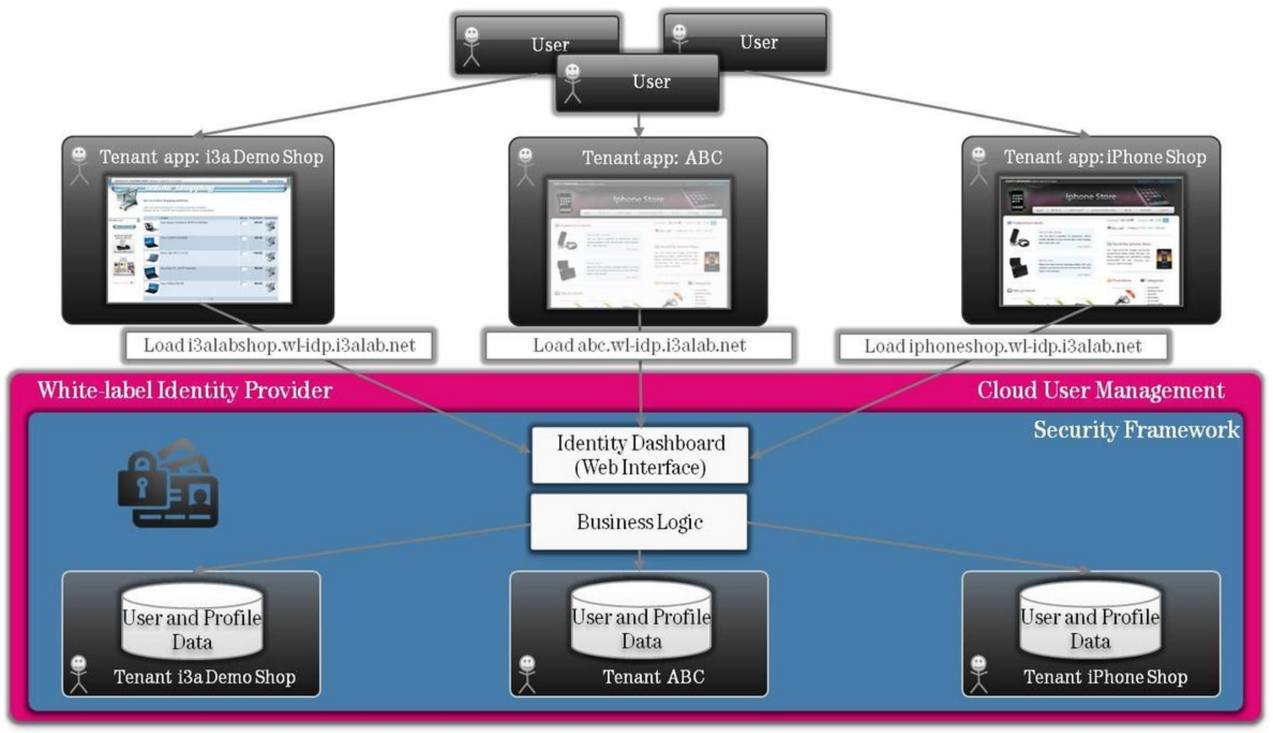
• Single sign-on between several Web services of one provider

• Self-administration for users (e.g., password reset and changes to user data)

• Ending of sessions

• Deletion of account

Moreover, Cloud User Management also offers the chance to log in or register users with the help of identities of popular Web services like Facebook, Google or other Open ID suppliers. The “social login” is implemented in cooperation with Janrain. The supplier of a Web application benefits not only from low operating costs, but also from the scalability and elasticity of the Cloud User Management. Sudden growth in the number of user registrations or massive increases in user registrations can be managed reliably. So the supplier does not have to buy in advance his own infrastructure to manage the user data, in order to cope with unforeseeable numbers of accesses.



WLIdP Overview

Brief Description of the service

The main features of the prototype are: User Life-Cycle Management - The WL-IdP offers tools for tenant administrators to support the handling of user life-cycle functions. It reduces the effort for account creation and management on the tenant side, as it supports the enforcement of policies and procedures for user registration, user profile management and the modification of user accounts. Tenant administrators can quickly configure customized pages for the inclusion of different authentication providers, registration of tenant applications with access to user profile data and the handling of error notifications.3rd Party Login - 3rd party login supports customers of the WL-IdP to enhance the reach of their websites by means of attracting users without forcing them to register new user accounts on their sites manually. 3rd party login allows users to register to the customers' sites with already existing digital identities from their favourite 3rd party identity providers, such as e.g. Google, Facebook or Yahoo. Thus, 3rd party login lowers the obstacles of registration processes and increases the number of successful business flows on the customers' sites. 3rd party login works as a proxy between a tenant's website and other well established 3rd party identity providers like the most important social networks (e.g. Facebook, MySpace), relevant Internet service providers (e.g. Google, Yahoo, Microsoft Live) and many OpenID identity providers Web Single Sign-On for Tenant Applications - As a tenant can configure several tenant applications that shall be linked to his WL-IdP instance, the main benefit for users is a single sign-on (SSO) to all these applications. Once a user has been successfully authenticated from the first application from a tenant, there is no need for further authentication and authorization for other linked tenant applications. Hosted User Profile Management - The WL-IdP offers a hosted user profile storage with tenant-specific user profile attributes. Developers do not have to run and manage their own persistent user data storages, but instead can use the WL-IdP's user profile storage as a SaaS offeringA user profile consists of two different sets of user attributes:

• Profile attributes managed by the user himself via a self-service Web interface

• Profile attributes used internally by the tenant applications.

These attributes are managed only by the tenant’s applications and are not visible in the user’s self-service Web interface. The tenant’s applications can access the full user profile via a secured REST interface.

Programming artifacts

The solution currently provides for all functions a WEB-Interface.

Technologies used

Java EE and HTTP

Runtime pre-requisites

Tomcat, WebDAV

IPR

Currently Deutsche Telekom holds no IPR’s. For Third party login the service of janrain® is used.

Publicly available documentation

http://www.developergarden.com/apis/apis-sdks/cloud-user-management

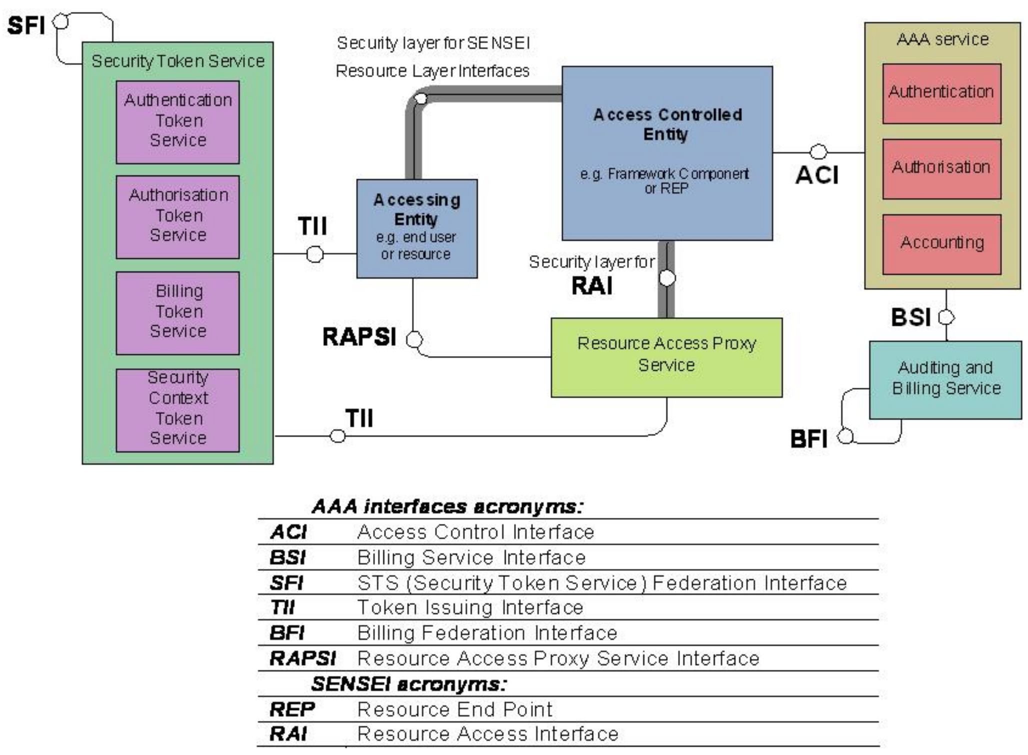
Access Control

Brief Description

An Access Control architecture for access to REST-based Web Services, including sensor and actuator resources. The asset consists of specifications of • service interfaces,

• interaction protocols

• ID/attribute/authorisation token formats.


SFI
Security Token Service
Authentication Token Service
Authorisation Token Service
Billing Token Service
Security Context Token Service
TII
Security layer for SENSEI Resource Layer Interfaces
Accessing Entity e.g. end user or resource
Access Controlled Entity
e.g. Framework Component or REP
ACI
AAA service
Authentication
Authorisation
Accounting
BSI
Security layer for RAI
RAPSI
Resource Access Proxy Service
Auditing and Billing Service
BFI
TII

AAA interfaces acronyms:	
ACI	Access Control Interface
BSI	Billing Service Interface
SFI	STS (Security Token Service) Federation Interface
TII	Token Issuing Interface
BFI	Billing Federation Interface
RAPSI	Resource Access Proxy Service Interface
SENSEI acronyms:	
REP	Resource End Point
RAI	Resource Access Interface

SENSEI Overview

The two parties being supported by the AAA services are the Accessing Entities (AE) and the Access Controlled Entities (ACE). Typically, an Accessing Entity

will authenticate itself with a Security Token Service and acquire appropriate tokens (authentication, authorization or context token) for the operations it

needs to perform. The Accessing Entity will present the tokens when making a service request to the Access Controlled Entity. The Access Controlled Entity

will pass the service request and token to the AAA function that is the decision point. The AAA function will return its decision to the Access Controlled

Entity and the original service request is either allowed or not. This is a very simplified explanation of the flow within the architecture – for a more

comprehensive description, refer to the document listed further below. The main objectives of the architecture are: • authentication of parties to each other within the system,

• authorisation of parties to do particular actions,

• logging and auditing of these actions,

• lightweight approach for low capability resources i.e. AE/ACEs

• capable of operating over multiple security domains i.e. federation

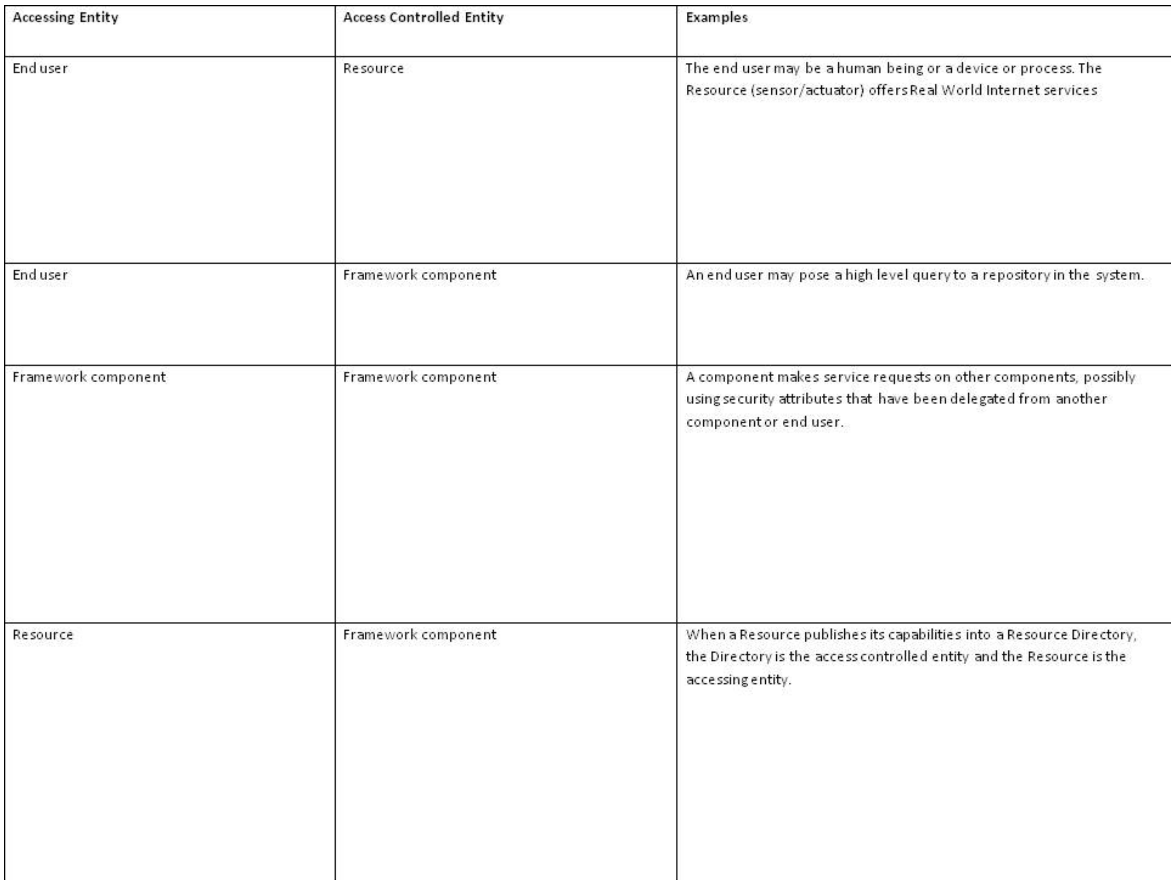
• support delegation of rights to allow actions to be carried out by others.

Brief Description of the service

The Access Control function has been designed as a service-oriented architecture. Both the Accessing Entities and Access Controlled Entities need to conform

to the interfaces specified, but these have been designed with resource-constrained platforms in mind. For example, the AAA decision point can be offloaded

from the Access Controlled Entity if appropriate. Below are some examples of how the Access Control service can be used.



Accessing Entity	Access Controlled Entity	Examples
End user	Resource	The end user may be a human being or a device or process. The Resource (sensor/actuator) offers Real World Internet services
End user	Framework component	An end user may pose a high level query to a repository in the system.
Framework component	Framework component	A component makes service requests on other components, possibly using security attributes that have been delegated from another component or end user.
Resource	Framework component	When a Resource publishes its capabilities into a Resource Directory, the Directory is the access controlled entity and the Resource is the accessing entity.

Stork Overview

The Access Control architecture addresses the need for lightweight access control for REST-based SOAs. It applies and integrates state-of-the-art in many

places, particularly SAML protocols and tokens, but extends it in others with introduction of new services and extensions to SAML protocols and token formats

designed specifically for optimising "Internet of Things" scenarios.

Technologies used

The Access Control is specified for use in REST-based systems.

Publicly available documentation

The Access Control asset was developed during the FP7 SENSEI project (www.sensei-project.eu/). The main reference report for this asset is D3.5 (Security and

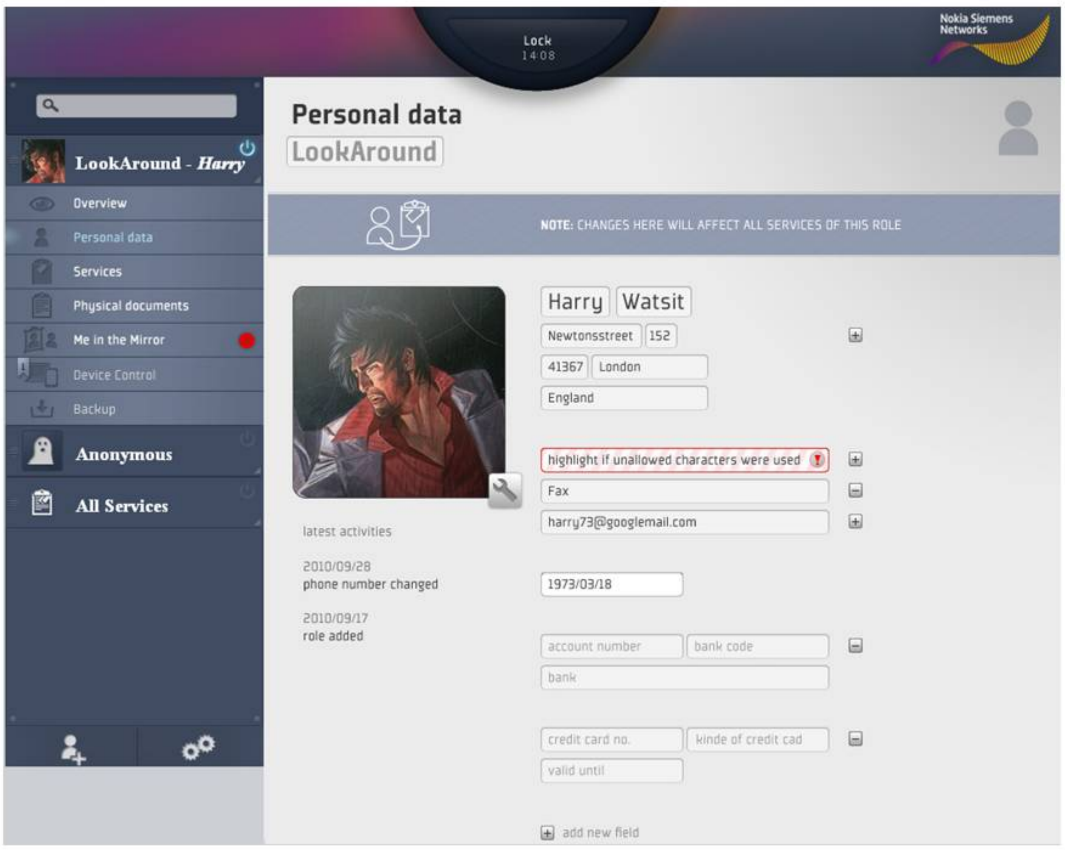
Accounting for SENSEI).

Privacy Enabler

Brief Description

The Privacy Enabler protects the end users personal data. Personal Data Portal enables end users being the master of his/her own data. This includes

administration of personal data and decision about the usage of data. It is based on the Identity Enabler.



NOTE: CHANGES HERE WILL AFFECT ALL SERVICES OF THIS ROLE

Harry Watsit
Newtonsstreet 152
41367 London
England
highlight if unallowed characters were used
Fax
harry73@googlemail.com
1973/03/18
account number | bank code
bank
credit card no. | kinde of credit cad
valid until
add new field

latest activities
2010/09/28
phone number changed
2010/09/17
role added

Privacy Enabler

Programming artifacts

Integration package for services providers with:

• Functional description

• Manual

Technologies used

The core of the platform is built with Java, and the interchanged messages are standard SAML 2.0 (to IDM) and OAuth 2.0 (to services).

Runtime pre-requisites

Standard Web2.0 complaint browser

IPR

No IPR, service will be hosted and is accessable via the internet.

Publicly available documentation

none

Idemix - Privacy-Preserving Credential Library

Brief Description

Description of the concept

We all increasingly use electronic services in our daily lives. To do so, we have no choice but to provide plenty of personal information for authorisation, billing purposes, or as part of the terms and conditions of service providers. Dispersing all these personal information erodes our privacy and puts us at risk of abuse of this information by criminals. Identity Mixer (idemix) allows users to minimise the personal data they have to reveal in such transactions. For instance, if electronic identity (eID) cards were realised with idemix, then teenagers possessing such eID cards could log onto a teenage chat room just proving that they are indeed 12-15 years of age without revealing any other information stored on the card such as their name or address.

Description of the service

The Identity Mixer – Credential Handling Library offers all the cryptographic algorithms to realise such anonymous authentication. This comprises the functionality for the issuer, client, and service provider. The library implements the credential system of Camenisch and Lysyanskaya [see http://idemix.wordpress.com for further information].
In addition to the basic credential system, the following additional features are currently supported for dealing with attributes contained in a credential when proving possession of credentials:

• Selective release of the attributes (minimal disclosure);

• Proving predicates over some of the attributes; and

• Verifiable encryption and anonymity revocation (useful for conditional anonymity) of some attributes

On top of that, the library also allows for proofs of possession of several credentials at the same time and to state various relations among the attributes contained in these credentials. For instance, as user could proof that she holds a driver's license as well as a student ID card and that she has made the driver's licence more than four years ago.

Programming artifacts

The library provides methods for system parameter generation, issuer key generation, user key generation, issuing of credentials, and proving ownership of credentials. There methods are parameterized with descriptions of their input parameters such as the issuer keys, the credential formats and the proof protocol format. These descriptions are done in XML. We refer to http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/for the specification of all methods and parameters.

Technologies used

The following technologies are used in the Identity Mixer – Credential Handling Library asset: JAVA.

Runtime pre-requisites

No direct pre-requisites. Being a library its needs some application logic that drives it. That is, to realise a typical application, these algorithms need to be embedded into an access control system, similarly as the algorithms to generate and verify, e.g., x.509 or SAML token. However, we also provide a policy language layer that allows the use of identity mixer credential in an access control system (See asset Identity Mixer – Credential-based Access Framework). The use of the Eclipse IDE development platform is recommended.

IPR

The asset is available under a license that grants all the right to use it free of charge (including commercial use).

See http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted

Publicly available documentation

Information available at:

- Documentation http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/
- PrimeLife website http://primelife.eu
- ABC4Trust website http://abc4trust.eu
- PRIME website http://prime-project.eu

Idemix - Credential-based Authentication Engine

Brief Description

Description of the concept

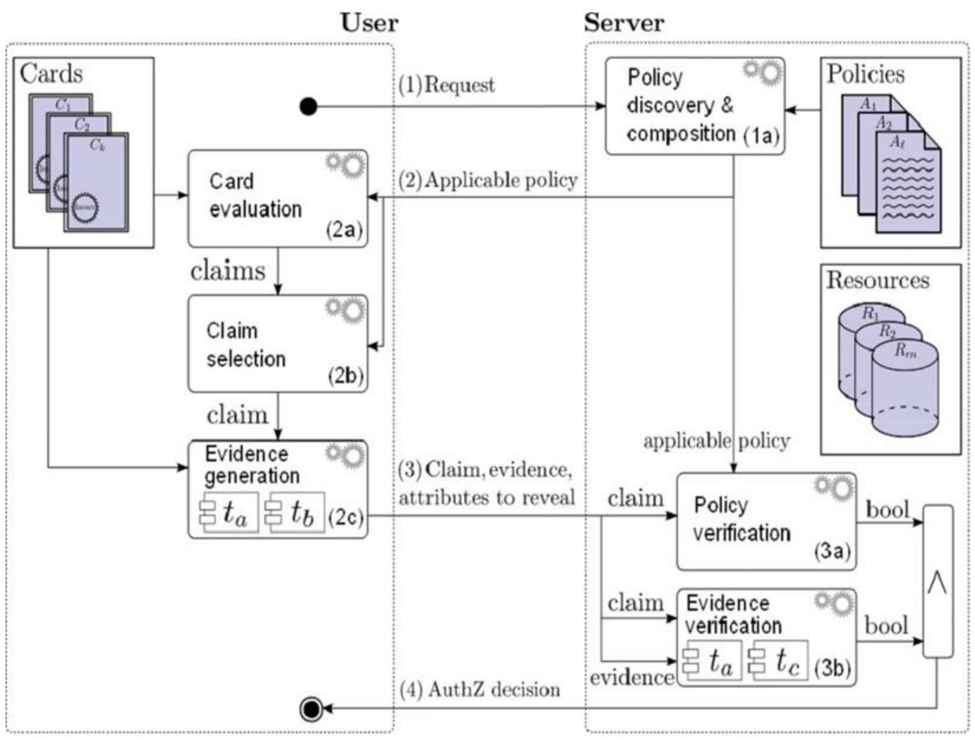
This software package offers an intuitive and flexible policy language and engine for access control based on credentials, sometimes also referred to as cards [CMNPS10]. A credential or card is a bundle of attributes issued by an issuer to a credential owner. By issuing the credential, the issuer vouches for the correctness of the attribute values contained in the credential with respect to the owner.

The policy language implemented in this package adheres to a data minimization principle, enabling relying parties to express which credentials a user has to present, which attributes of those credentials have to be revealed, and which conditions the attributes have to satisfy. The language is based on the card-based access requirements language (CARL) [CMNPS10], supporting advanced features such as selective revealing of attributes, policies involving multiple cards, and proving conditions involving attributes from different cards. The package also defines a corresponding claims language, by means of which the user communicates to the relying party the actual identity properties being proved.

The policy language is inherently technology-independent, in the sense that any authentication technology can be used to substantiate the authenticity of the claims made by a user. The language is particularly targeted, however, at leveraging the privacy-enhancing features of anonymous credentials. The current implementation of the engine offers support for Identity Mixer credentials , but other (anonymous or non-anonymous) credential mechanisms can be added. The policy language makes it much easier for relying parties to support Identity Mixer credentials as a valid means of authentication and write their own access policies matching their specific needs.

Description of the service

The policy engine implements most of the steps in the typical authentication interaction. Initially, the user contacts a server to request access to a resource (1). Having received the request, the server responds with the applicable access control policy for the resource (2). The applicable policy contains the card requirements expressing which cards have to be presented, which attributes on those cards have to be revealed, which conditions the attributes on the cards have to satisfy, and which entities are considered trusted issuers for these cards. Upon receiving the policy, the user's system searches the user's card store for combinations of cards that fulfill the policy (2a) and for each combination, derives the claim that will be revealed if this combination is chosen. Once the user indicates (2b) which combination of cards is to be used for the authentication, the cryptographic evidence for the chosen claim is generated (2c). The claim and the evidence are sent to the server (3). Finally, the server verifies whether the claim satisfies the policy (3a) and whether the cryptographic evidence supports the claim (3b). If so, access to the resource is granted (4). The current policy engine implements steps (2a), (2c), (3a), (3b) in the above scenario. Meaning, given an access control policy expressed in the policy language (2), the package parses a user's credential store and returns the list of card combinations and associated claims that can be used to satisfy the policy (2b). For a particular chosen claim, the engine invokes the Identity Mixer library and generates an Identity Mixer token supporting the chosen claim (2c). It also implements the server's functionality of checking whether the transmitted claim fulfills the required policy (3a) and verifying that the Identity Mixer token supports the claim (3b). The policy engine does not, however, provide a user interface to facilitate the claim selection process in step (3b), and neither does it implement the policy discovery step (1a). For the former, an existing identity selection based on the open-source project Higgins or on Microsoft's Cardspace would have to be integrated with the engine. For the latter, one could use any access control engine, e.g., based on XACML.



Programming artifacts

The library provides methods for:

• given a server's access control policy and a user's credential store, determine which combinations of credentials can be used to satisfy the policy, and what the generated claim would be for each of these combinations;

• given a selected claim and a user's credential store, generate the cryptographic evidence (by calling the Identity Mixer Library) to substantiate the authenticity of the claim and include the evidence in the claim;

• given a server's policy and a user's claim, verify that the provided claim satsifies the policy;

• given a user's claim and corresponding cryptographic evidence, verify that the evidence substantiates the authenticity of the claim (by callling the Identity Mixer Library).

Technologies used

The Identity Mixer – Credential-based Policy Engine is implemented in Java. It uses the Eclipse Xtext package and the Xtext Type System for parsing and manipulating the policy and claims languages. It also uses the Identity Mixer – Credential Handling Library for generating the cryptographic evidence for claims.

Runtime pre-requisites

To be deployed in practice, this library needs to be embedded in an access control system that can determine the applicable policy for a requested resource. Also, on the user's side, an identity selection user interface is needed to let the user choose which combination of credentials to use when there are multiple possibilities. The use of the Eclipse IDE is recommended to easily load the Xtext package on which it depends.

IPR

The asset is available under the EPL (Eclipse Public License).

Publicly available documentation

Information available at: [CMNPS10] Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-

preserving access control.

In J. Joshi and B. Carminati, editors, 15th ACM Symposium on Access Control Models and Technologies - SACMAT 2010, pages 119-128. ACM, 2010.

Accountable privacy policies

Brief Description

Beyond the definition of precise privacy policies, another strong requirement (which should be included in the revision of the EU Directive 95/46/EC) is the accountability of the data controllers: as stated in the Article 29 Working Party Opinion 3/2010, accountability means that controllers should be able to “show how responsibility is exercised and make this verifiable. Responsibility and accountability are two sides of the same coin and both essential elements of good governance. Only when responsibility is demonstrated as working effectively in practice can sufficient trust be developed.” A key design choice to implement accountability is the log architecture itself. Indeed, recording log entries on a device controlled by an actor who may be involved in a claim for which this log would be used as evidence may not be acceptable for the other parties. To address this issue, we have introduced a framework for the specification of log architectures and proposed criteria to characterize “acceptable log architectures”. These criteria depend on the functional architecture of the system itself and the potential claims between the parties. They can be used to check that a log architecture is appropriate for a given set of potential claims and to suggest improvements to derive an acceptable log architecture from a non-acceptable log architecture. On the formal side, we have shown that, for a given threat model, the logs produced by acceptable log architectures can be trusted as evidence for the determination of liabilities: technically speaking, any conclusive evaluation of a claim based on these logs produces the same verdict as the evaluation of the claim based on the sequence of real events.

Technologies used

Trace semantics of log architectures.

Runtime pre-requisites

No run-time pre-requisite.

IPR

Publicly available

Publicly available documentation

D. LE MÉTAYER, E. MAZZA, M.-L. POTET. Designing log architectures for legal evidence, in "8th International Conference on Software Engineering and Formal Methods, SEFM’2010", IEEE, 2010, p. 156-165.

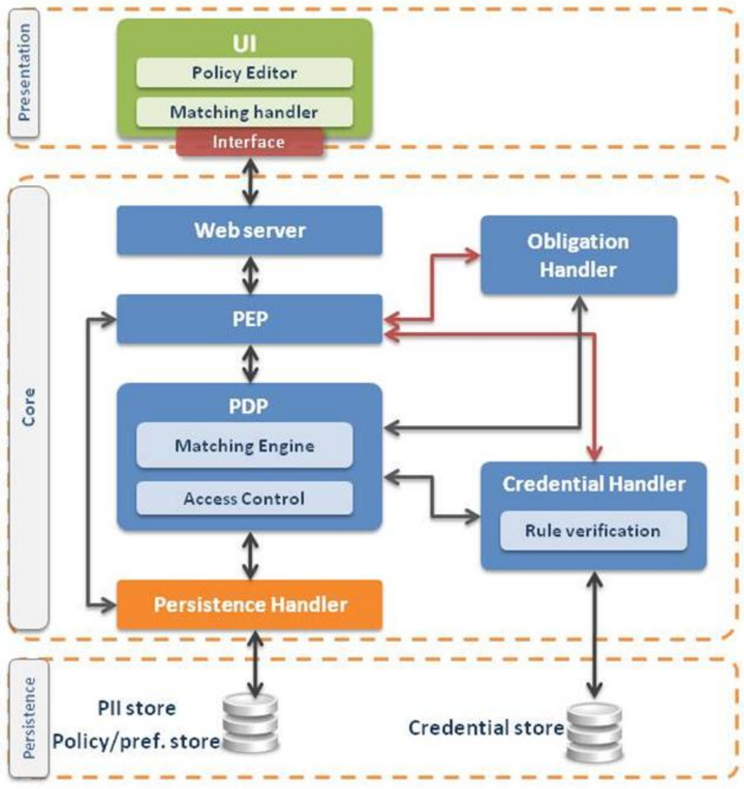
PrimeLife Policy Engine: PPL

Brief Description

The PPL language [1] is specified as an extension of the XACML (eXtensible Access Control Markup Language) [2] language, then the PPL engine is designed to run together with the HERAS-XACML engine [3] (that only handles XACML access control rules). The architecture chosen for the deployment of the PPL engine is symmetric because Data Subjects and Data Controllers (have similar requirements: deciding whether a given personal identifiable information (PII) can be shared with a data controller (resp. third party); handling obligations associated with data; storing data and associated preferences (resp. sticky policies). Using the same architecture everywhere to handle scenarios where one party can have multiple roles (e.g. collecting data and next disclosing it to third parties). The PPL engine executes multiple tasks in an automated way like: enforcing access control policies, generating and verifying cryptographic proofs related to credential requirements, matching between data handling preferences and data handling policies, generating and enforcing sticky policies, checking authorization, controlling the downstream usage of data, handling obligations.

PPL Engine Architecture

The entire architecture of the PPL engine (see Figure X) can be represented by 3 layers architecture. The first one presents the user interface layer. The second, the Core layer, represent the main elements of the PPL Engine, which is composed by different subcomponent that we will describe their function below. And the last one represents the persistence layer that is in charge of storing the different data and policies that are used during a transaction between the DC and DS.



Presentation Layer

The presentation layer is responsible of the display to the end user. The presentation layer contains two types of components: the Policy editor that displays and provides a way to manage all the information related to the DS, DC and the third party. The presentation payer offers a restful and a Web Service API in order to use directly the engine.

Core Layer

The Core layer is composed in four main components that are implementing the new concepts introduced within PrimeLife. These components are: • Policy enforcement point (PEP): This component formats then dispatches the messages to the corresponding component according to the state of the execution process. The decision made by the PDP is enforced in the PEP, meaning that if the PDP decided to provide a data or enforce the access of one resource, this data/resource is collected, formatted and sent to the receiver through the PEP. • Policy Decision Point (PDP): it is the core of the PPL engine. All the decisions are taken in this component. This latter has two functionalities: the Matching engine that matches between the preferences of the DS and the privacy policy of the DC. The matching is done to verify if the intentions of the data controller in terms of private data usage are compliant with the data subject preferences. The Access control engine is in charge of the enforcement of the access control rules related to the local resources. It analyses the resource query, check the access control policy of the requested resource and decides whether or not requester satisfies the rule. • Credential Handler: one of the new features introduced in PPL is the support of the credential based access control. This feature is implemented by the credential handler that manages the collection of credential held by an entity, selects the appropriate credentials in order to generate a cryptographic proof and verifies the cryptographic proofs of the claims received from external entities. The credential handler component contains the subcomponent Rule Verification; the PPL policy contains a description of the credential requirements (for access control), the Rule Verification component evaluates whether the claim provided by a user that wants to access a resource satisfies the credential based access control rule. The credential Handler refers to the IBM asset. • Obligation handler: is responsible for enforcing the obligations that have to be satisfied by the DC. This engine executes two main tasks: setup the triggers related to the actions required by the privacy preferences of the Data Subject, and executes the actions specified by the data subject whenever it is required.

Persistence Layer

The persistence layer is represented by the Data/Policy store contains all the information on private data and their related policies and by the credential store which contains all the credentials and the certified information held by an entity. The access to this store is exclusively allowed to the Credentail Handler component.

Methods and parameters

To expose the engine functionality to allow other developers reuse our engine, we decided to create the API for the management of data subject and data controller instances. The interface is prepared in form of the RESTful HTTP methods.

Data Subject API

For managing PIIs and PreferenceGroups on the data subject we provide the following API: Managing PII • Create a PII PUT /api/pii?name=attributeName&value=attributeValue HTTP/1.1 Parameters: o name - attribute name of the PII (the type). o value - attribute value of the PII. • Update a PII POST /api/pii?name=attributeName&value=attributeValue HTTP/1.1 Updates the value of the PII with that attribute name. Parameters: o name - attribute name of the PII (the type). o value - new attribute value of the PII. • Delete a PII DELETE /api/pii?name=attributeName HTTP/1.1 Parameters: o name - attribute name of the PII (the type). • Get all PIIs GET /api/pii HTTP/1.1 Returns list of all PIIs stored in the database. Managing Preference Groups • Create a preference group PUT /api/groups HTTP/1.1 Creates one or more preference groups. Request content: String representation of the resource policy in the XML format. The root element of the policy document must be http://www.primelife.eu/ppl:Policy or http://www.primelife.eu/ppl:PolicySet.

• Update a preference group POST /api/groups/group id[?new pref group=new pref group id] HTTP/1.1 Updates the preference group by merging the mismatches from the sticky policy, so the next time when the matching is performed (between the policy from specified preference group and the same the data controller's resource policy) there will be no mismatch. Parameters: o group id - the policySetId or policyId of the preference group. o new pref group - (optional) identifier of the new preference group if the result shall be stored as a new preference group. o policy { the policy of the resource the user is trying to gain access to. Note: this is the same input as given to the accessResource() method.

• Delete a preference group DELETE /api/groups/pref group id HTTP/1.1 Deletes a preference group. Parameters: o pref group id - the identifier of a preference group.

• Get all preference groups identifiers GET /api/groups HTTP/1.1 Gets a list of all preference group names stored in the database. Response: A JSON list: [PrefGroupName1, PrefGroupName2]

• Get a preference group policy GET /api/groups/pref group id HTTP/1.1 Gets the XML of a specific preference group. Parameters: o pref_group_id - the identifier of a preference group

Data Controller API

Direct management of data controller's resource and PII store is now possible by calling RESTful interface methods. • Uploading resource data and policy PUT /api/resource/name HTTP/1.1 Parameters: o Name – resourse name Request content: String representation of the resource policy in the XML format. The root element of the policy document must be urn:oasis:names:tc:SAML:2.0:assertion:Assertion. Response: Resource id • Uploading PII PUT /api/pii HTTP/1.1 Parameters: o Name - PII's attribute name (e.g. http://www.w3.org/2006/vcard/ns#email). o Value - PII's attribute value (e.g. john.doe@example.com). Request Content: String representation of the sticky policy in the XML format. The root element of the sticky policy document must be http://www.primelife.eu/ppl/stickypolicy:Attribute. Response: PIIid • PII downstream usage request for a single PII POST /api/pii HTTP/1.1 Request content: String representation of the request policy in the XML format. The root element of the policy document must be urn:oasis:names:tc:SAML:2.0:assertion:Assertion. Response String representation of the claims in the XML format. The root element of the document is http://www.primelife.eu/ppl/claims:Claims.

Technologies used

- JAVA J2EE
- MySQL
- Jetty server
- Apache Tomcat
- Hibernate
- JAXB
- HyperJAXB
- HERAS (XAML java engine)

Runtime pre-requisites

The Enabler is deployed as a service, it is designed to be deployed in an APACHE server and a MySQL Database.

IPR

The code source is restricted under SAP Licence.

Publicly available documentation

[1] 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.

[2] Moses, T.: OASIS eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard oasis-access control-xacml-2.0-core-spec-os, OASIS (February 2005)

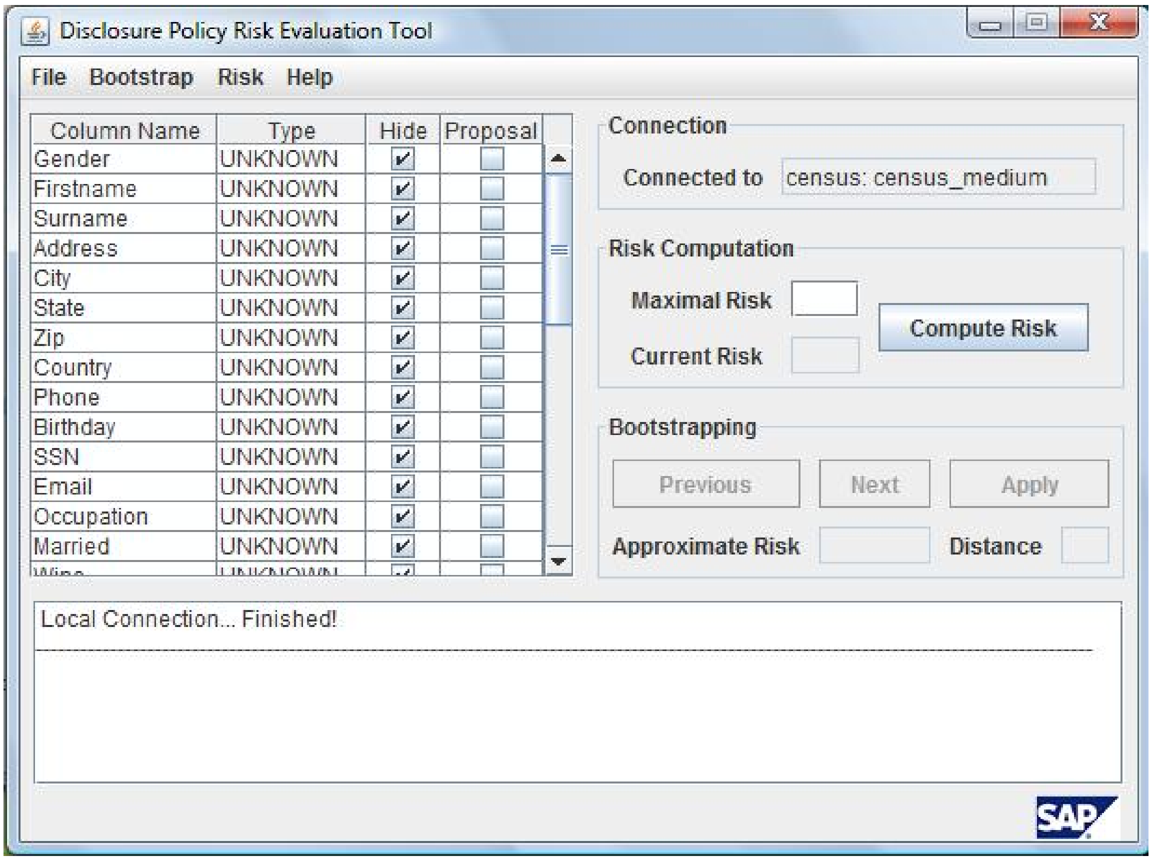
[3] Holistic Enterprise-Ready Application Security http://www.herasaf.org/heras-af-xacml.html

[4] PrimeLife website http://primelife.eu

Database Anonymization Optional Asset

Brief Description

Many companies have to share various types of information containing private data without being aware about the threats related to such non-controlled disclosure. Therefore we propose a solution to support these companies to evaluate the disclosure risk for all their types of data; by recommending the safest configurations using a smart bootstrapping system. We propose a tool that can be used to automate this process, providing 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.



Column Name	Type	Hide	Proposal
Gender	UNKNOWN	✓	☐
Firstname	UNKNOWN	✓	☐
Surname	UNKNOWN	✓	☐
Address	UNKNOWN	✓	☐
City	UNKNOWN	✓	☐
State	UNKNOWN	✓	☐
Zip	UNKNOWN	✓	☐
Country	UNKNOWN	✓	☐
Phone	UNKNOWN	✓	☐
Birthday	UNKNOWN	✓	☐
SSN	UNKNOWN	✓	☐
Email	UNKNOWN	✓	☐
Occupation	UNKNOWN	✓	☐
Married	UNKNOWN	✓	☐

Connection

Connected to: census: census_medium

Risk Computation

Maximal Risk:

Current Risk:

Bootstrapping

Approximate Risk: Distance:

```
Local Connection... Finished!
------------------------------------------------------------------------------------------------
```

The service that can provide this asset is a remote estimation of the risk for an anonymized database to be re-identified partially. If a database owner wants to share his DB with a non trusted third party, there is always a risk that some elements of his DB can give an indication about some other hidden elements. This is due to the possibility of aggregating different sources. Using this service the DB owner can evaluate the risk of his shared resource and hide the risky elements.

Programming artifacts

Integration package for services providers with:

- Functional description
- Interface description

Technologies used

- Java J2EE
- MySQL
- Apache Server

Runtime pre-requisites

- Java J2EE
- MySQL
- Apache Server

IPR

Closed source SAP Licence

Publicly available documentation

- FP7 EU PrimeLife Project [1]
- Slim Trabelsi, Vincent Salzgeber, Michele Bezzi and Gilles Montagnon, "Data Disclosure Risk Evaluation", CRISIS 2009, The 4th IEEE International Conference on Risks and Security of Internet and Systems, September 2009 , Toulouse, France.
- Michele Bezzi and Montagnon Gilles and Vincent Salzgeber and Slim Trabelsi, "Sharing Data for Public Security", Privacy and Identity Management for Life(2010), 5th IFIP PrimeLife International Summer School, Nice, France, September 7-11, 2009

References

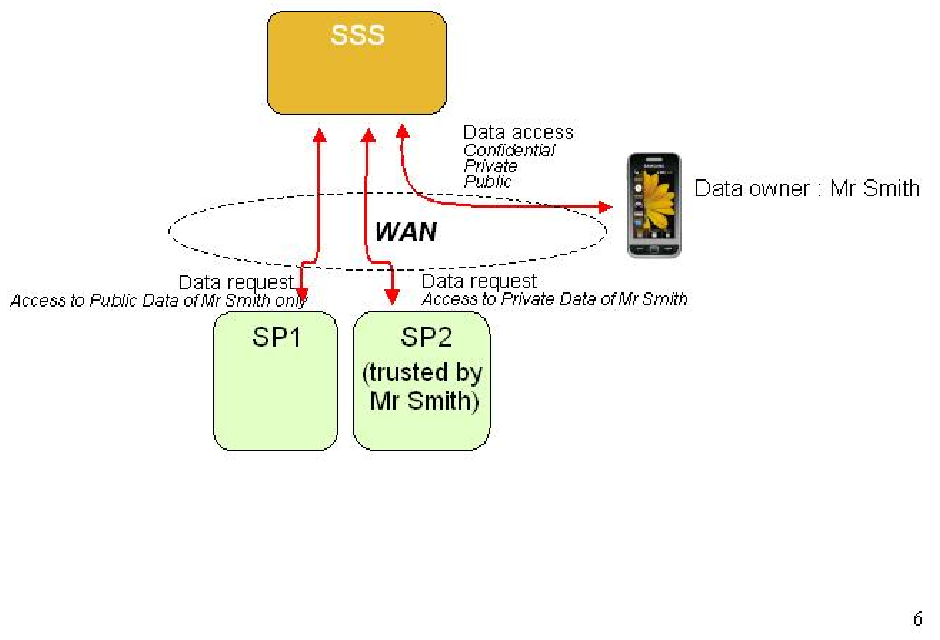
[1] http://www.primelife.eu-http://www.primelife.eu/

Secure Storage Service (SSS) Optional Asset (Thales)

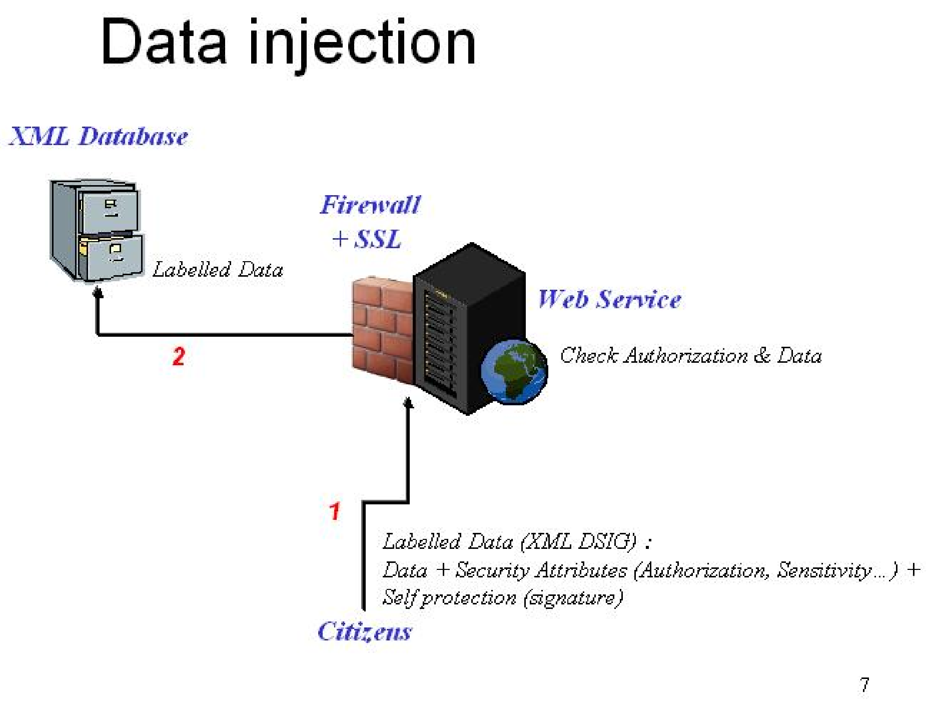
Brief description

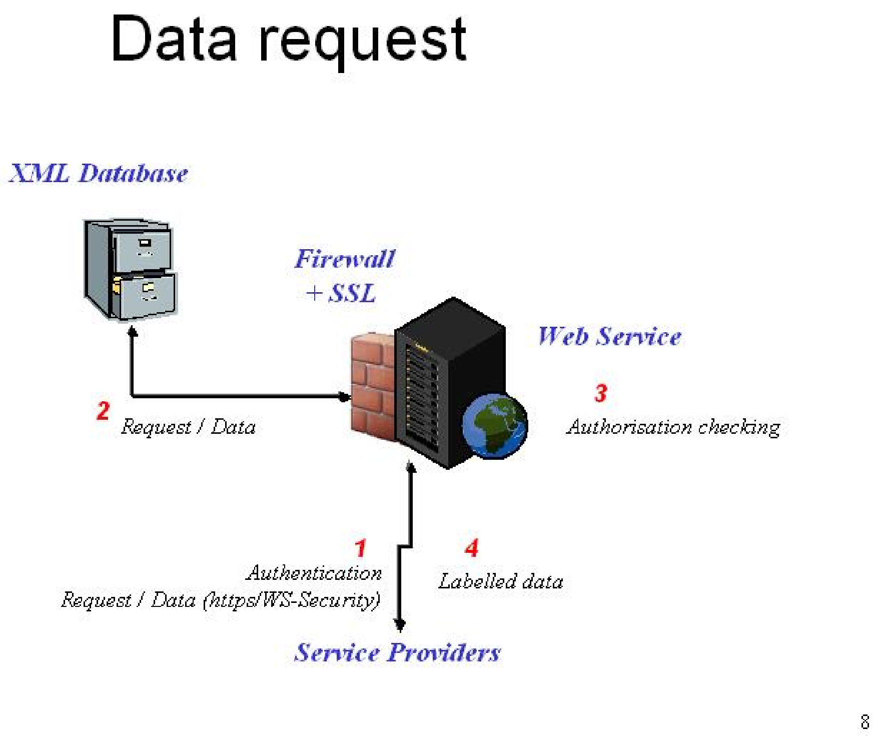
The main objectives of SSS is to provide :

- a secure storage to sensitive / private data
- privacy-oriented capacities, according to the UE legislation


SSS
Data access
Confidential
Private
Public
Data owner : Mr Smith
WAN
Data request
Access to Public Data of Mr Smith only
Data request
Access to Private Data of Mr Smith
SP1
SP2 (trusted by Mr Smith)
6

Description of the service


Data injection
XML Database
Labelled Data
Firewall + SSL
Web Service
Check Authorization & Data
2
1
Labelled Data (XML DSIG) :
Data + Security Attributes (Authorization, Sensitivity…) +
Self protection (signature)
Citizens
7



Programming artifacts

Functions available for the user :

- Data injection
- Data update
- Data deactivation
- Get the list of SPs who had access to the profile and when, and what usage they did of the data

Functions available for the SPs :

- Data request
- Security Attribute request

NB : These functions are defined according to European rules : 95/46/CE, 2002/58/CE, COM(2010) 609

Technologies used

- XML DSIG
- Web Services

Runtime pre-requisites

To be completed

IPR

Closed source

Publicly available documentation

http://www.smarturbanspaces.org/links-docs

Morphus Optional Asset (INRIA)

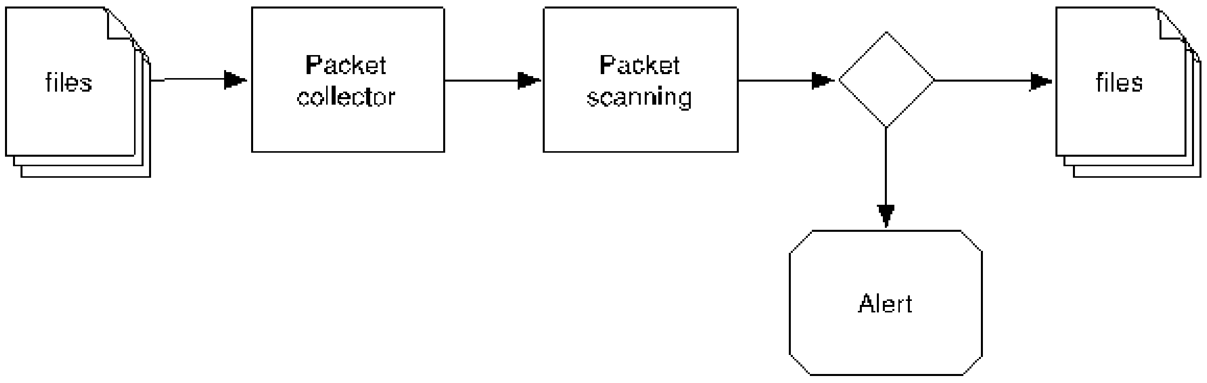
Brief Description

Morphus is a generic malware detector based on graph signatures. Morphus discriminates binary codes with respect to some signature database. 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.

Programming artifacts

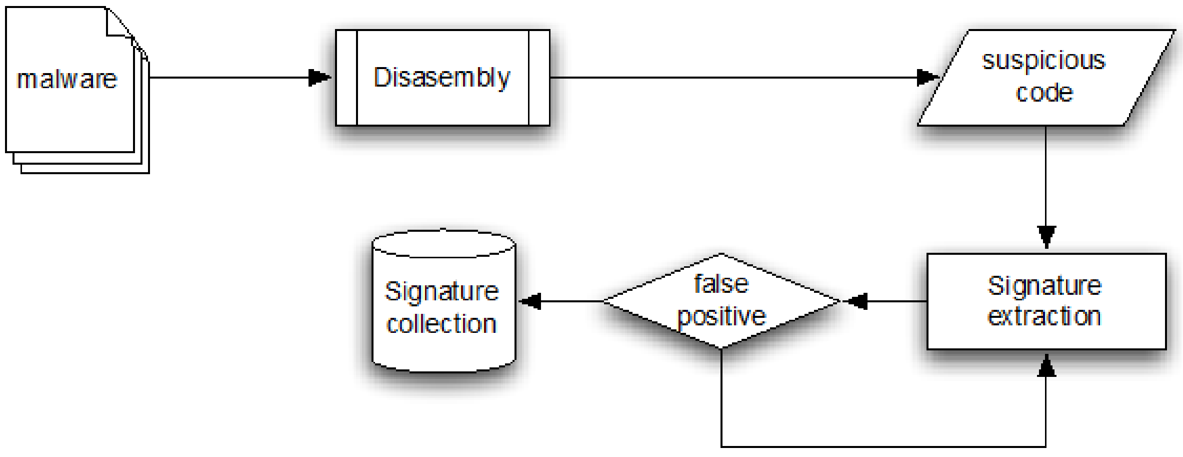
Program offered to end-users:

-Scanning. Given a binary code and a database, the output of scanning is a log file containing a distance vector summarizing the similarities of the input file with respect to the database.



Program offered to administrator:

-Learning. Given a binary code and a database, learning adds the signature of the code to the database and outputs a log-file summarizing the quality of the learning analysis (description of inputs, number of scanned instructions, comparison to other malware, modification of the database)



Technologies Used

- C
- beaengine
- PIN

IPR

To be completed.

Publicly available documentation

- http://lhs.loria.fr/index.php?option=com_content&view=article&id=95&Itemid=75
- [1] Jean-Yves Marion, Daniel Reynaud. Dynamic Binary Instrumentation for Deobfuscation and Unpacking, presentation at the In-Depth Security Conference Europe 2009
- [2] Guillaume Bonfante, Matthieu Kaczmarek, Jean-Yves Marion. Architecture of a Morphological Malware Detector, Journal in Computer Virology 5, 3 (2009).

Materializing the Interface to Networks and Devices (I2ND) in FI-WARE

Introduction

Following is a description of the assets that have been adopted as baseline for building a reference implementations of the GEs in the Interface to Networks and Devices (I2ND) chapter of FI-WARE. The reference implementation of a Generic Enabler is typically based on the evolution and integration of a number of assets, some being open source, therefore publicly available, while others being provided by partners of the FI-WARE project. A Backlog of Themes, Epics, Features and User-Stories followed for the evolution and integration of assets linked to the reference implementation of a Generic Enabler is also included.

Finally, a list of topics still being addressed at a high level follows the description of assets in this chapter. They are mapped into Themes and Epics in the Chapter Backlog. Features and User-Stories, derived from refined of these Theme and Epics will be allocated to Backlogs linked to GEs in the future.

For a comprehensive vision on how Interfaces to Networks and Devices are addressed in FI-WARE, you can go here. We highly recommend you to learn about the vision before analyzing how reference implementations of GEs are being materialized.

Connected Device Interfacing

Baseline Assets

- WEB Runtime Engine is an environment to host widgets exploiting interfaces to connected device capabilities. it is based on WAC APIs, and it is developed for Android platforms.
- Meego is a operating system designed for mobile devices, and can be extended to provide the necessary programming interfaces for WAC / PhoneGap APIs.
- Web API is an implementation and framework for providing access to low level device functionality from inside a web browser. Such functionality is provided via a Javascript interface to hosted websites.
- Reinforcement Learning is an algortihm for end-user QoE evaluation.

Epics

- FIWARE.Epic.I2ND.CDI.QoE
- FIWARE.Epic.I2ND.CDI.DeviceConnectivity
- FIWARE.Epic.I2ND.CDI.DeviceFeatures
- FIWARE.Epic.I2ND.CDI.DeviceSensors
- FIWARE.Epic.I2ND.CDI.PersonalDataServices
- FIWARE.Epic.I2ND.CDI.UserProfile
- FIWARE.Epic.I2ND.CDI.MediaServices
- FIWARE.Epic.I2ND.CDI.Phone
- FIWARE.Epic.I2ND.CDI.Messaging
- FIWARE.Epic.I2ND.CDI.RemoteManagement

Features

- FIWARE.Feature.I2ND.CDI.DeviceFeatures.CommonDeviceInterface
- FIWARE.Feature.I2ND.CDI.DeviceFeatures.CapabilityDescription
- FIWARE.Feature.I2ND.CDI.Phone.DiscoverPhoneFunctionality
- FIWARE.Feature.I2ND.CDI.Phone.UtilisePhoneFunctionality

User Stories

- FIWARE.Story.I2ND.CDI.DeviceFeatures.CommonDeviceInterface.FrameWorkAndPlatform
- FIWARE.Story.I2ND.CDI.DeviceFeatures.CommonDeviceInterface.WorkingDevelopmentEnvironment
- FIWARE.Story.I2ND.CDI.DeviceFeatures.CapabilityDescription.CapabilityDetectionViaAPI
- FIWARE.Story.I2ND.CDI.DeviceFeatures.CapabilityDescription.HTTPCapabilityDescription

Cloud Edge

Baseline Assets

- The Network Organizer is a piece of software which realizes the unification of the access to the content in the home system at the file system level.
- The ZigBee GAL(Gateway Abstraction Layer) is a C++ software which implements the specifications of the ZigBee Gateway and provides a proper

abstraction of ZigBee devices and ZigBee networks.

Epics

- FIWARE.Epic.I2ND.CE.HomeSystemManagement
- FIWARE.Epic.I2ND.CE.HSModelling
- FIWARE.Epic.I2ND.CE.OtherDrivers

Features

- FIWARE.Feature.I2ND.CE.HomeResourceAbstractionLayerFirstRelease

Network Information and Control

Baseline Assets

- NOX based OpenFlow controller is a system used to control OpenFlow-enabled network devices.
- Network Element Virtualizer the glue technology between network resources and the cloud infrastructure.
- DRAGON a GMPLS-based control plane.

Epics

- FIWARE.Epic.I2ND.NetIC.VNPSetup
- FIWARE.Epic.I2ND.NetIC.VNPConnect
- FIWARE.Epic.I2ND.NetIC.VNPSchedule
- FIWARE.Epic.I2ND.NetIC.VNPGreen
- FIWARE.Epic.I2ND.NetIC.VNPRecover
- FIWARE.Epic.I2ND.NetIC.NetworkStatistics
- FIWARE.Epic.I2ND.NetIC.NetworkResourceControl

Features

- FIWARE.Feature.I2ND.NetIC.NetworkResourceControl.EquipmentConfiguration
- FIWARE.Feature.I2ND.NetIC.VNPSchedule.PathCalculation&Reservation&Release

User Stories

- FIWARE.Story.I2ND.NetIC.NetworkResourceControl.EquipmentConfiguration.EquipmentProvisioning

Service, Capability, Connectivity and Control

Baseline Assets

- OpenEPC Platform is a software implementation inspired by the innovative all-IP aspects of the 3GPP Evolved Packet Core (EPC) enabling the prototyping of features like QoS and charging, mobility management, access and

security for 3GPP and non-3GPP wireless technologies.
- Device Connection Platform provides device accessibility.
- API mediation The API mediation component provides a set of APIs to several actors interacting API providers. It deals with non functional properties of API exposure, such as authorization and accounting.
- Fast Telecom Service helper It makes it possible to service providers to create easily telecom services.
- Voicemail enabler It makes it possible for a service provider to create voice based services.
- Intelligent Network Identity Management enables intelligent ID management for multiple devices belongig to a user.
- Network-centric Resource Management enables technology-independent control over network resources both in wireless and wired network domains.
- Seamless Network Connectivity enables best connectivity for multi interface terminals incl. interface selection per flow and reservation of resources within the network.

Epics

- FIWARE.Epic.I2ND.S3C.ExposedNetworkInformation
- FIWARE.Epic.I2ND.S3C.CoreNetworkControl
- FIWARE.Epic.I2ND.S3C.CoreNetworkDeviceManagement
- FIWARE.Epic.I2ND.S3C.NetworkLevelDataDelivery
- FIWARE.Epic.I2ND.S3C.SMS+MMS ENABLERS
- FIWARE.Epic.I2ND.S3C.Telecom platform 1
- FIWARE.Epic.I2ND.S3C.API_mediation_1
- FIWARE.Epic.I2ND.S3C.WSC 1
- FIWARE.Epic.I2ND.S3C.DeviceReachability

Topics still being addressed at high-level

Epics

- FIWARE.Epic.I2ND.CDI.Security.SubscriberPrivacyProtection

WEB Runtime Engine

Brief Description

The WEB Runtime Engine on Android is an asset materialising the use of interfaces to device features for widgets. This is currently implementing a part of WAC APIs, as well as some additional feature like W3C Widget P&C and side loading of widgets thought SD card.

Programming artifacts

The WEB Runtime Engine implies the extension of JavaScript APIs by providing a new 'object' for WAC APIs, which could be properly interpreted by JavaScript engine together with its attributes and methods. The web applications can get access to device resources, so the new JavaScript object has to be granted access to these resources while no JavaScript objects commonly have. Such access permissions must be granted or revoked according to configurable security policies: in other words, the link between the JavaScript world and the device resources has to be filtered so that proper actions can be executed. Being Android based on Java, while Javascript is provided to the applications, interactions between Java and JavaScript environments take place in different situations:

- At startup (document loading)
- At module loading
- Every time a module method is accessed.

Technologies used

The WEB Runtime Engine aims to be installable in post sale mode so it relies on official Android SDK and APIs without any modification of the source code (e.g. WebKit source code) neither installations of native browser widget engine (possible but not part of Android official support/roadmap). The WEB Runtime Engine operates on Android OS platform, and it is developed exploiting the Android SDK. A Javascript wrapper has been created for every Java object, in order to filter and properly convert data types on parameters' passages.

Runtime pre-requisites

Widget development follows W3C standard rules.

Known software requirements

The WEB Runtime Engine runs on Android platforms 2.1.1/2.1.2

IPR

This asset will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

The WEB Runtime Engine is not (at the moment) publicly available

Meego

Brief Description

Meego is a Linux-based, open source software platform, it is designed to run on a wide range of devices, including netbooks, handheld computing, mobile phones, in-vehicle infotainment devices, smart TVs, tablets, etc.

Programming artifacts

As a Linux-based operating system MeeGo provides a wide range of programming interfaces. The primary interface MeeGo provides is a unifrom QT [1] based API.

Technologies used

- Linux
  - RPM repositories
- C++
- QT
- Evolution (Email, calendar)
- Empathy (IM)
- Gwibber (microblogging)
- Chromium / WebKit (browser)
- Banshee (multimedia player)

Supporting Programming Libs

- QT
- GTK+

Runtime pre-requisites

Current binary released of MeeGo support Intel Atom based notebooks, and Intel Atom based In-Vehicle Entertainment Platforms.

Further details on the supported versions of MeeGo can be found on the Core Software Platform web page: https://www.meego.com/downloads/releases/1.2/meego-v1.2-core-software-platform

Known software requirements

None.
MeeGo is an full Linux operating system (distribution) and therefor does not dictate any pre-existing software requirements.

IPR

MeeGo is a distribution containing a number of software packages all distributed under an OSI Open Source compatible license. MeeGo is composed of two distinct blocks:

- **The MeeGo Operating System Software**: The majority of packages contained in the operating system are licensed under OSI compatible licenses, where code modifications should be open sourced.
- **The MeeGo User Experience Subsystem Software**: The majority of packages contained in the subsystem are licensed under a “permissive OSI compatible license” – a BSD-style license which does not mandate code modifications are open sourced.

Further information on the MeeGo Licensing Policy can be found on the MeeGo Licensing Policy page. https:// www.meego.com/about/licensing-policy

Publicly available documentation

Further documentation is available both on MeeGo and QT at the following locations:

- QT [1]
- MeeGo [2]

References

[1] http://qt.nokia.com/products/
[2] https://www.meego.com/

Web API

Brief Description

The WebAPI is a JavaScript extension which allows web developer to create client side JavaScript code to access low level client side system details.
It is implemented via the Intel Web Connector browser plugin. This is a standard NPAPI based browser plugin which is automatically extensible; additional APIs and functionality are added to the connector at run time. When the JavaScript developer invokes the plugin and “requests” a functional element, the native code for the functional element is retrieved from an Intel server and executed on the client.

Programming artifacts

The WebAPI provides a set of JavaScript interfaces for web developers. Internally the WebAPI is implemented as a browser plugin, using native code.

Technologies used

- JavaScript
- Native System APIs on
  - Windows
  - Linux
  - MacOS
- Browser plugin Interface (NPAPI)

Runtime pre-requisites

Web developer, implementations use standard w3C implementation tools, ECMA script, HTML, CSS, etc.

Known software requirements

The WebAPI currently runs on the following operating systems and browsers:

- Windows
  - Firefox 3.5x, 3.6x
  - Chrome 4.0x
  - Opera 10.x
  - Internet Explorer 8.x
- Linux
  - Work in progress on:
    - Firfox
    - Chrome
    - Safari
- MacOS
  - Work in progress on:
    - Firfox
    - Chrome
    - Safari

IPR

The WebAPI (Intel Web Connector, the additional functional blocks, and the JavaScript sample source code) are all distributed under the Intel Sample Source Code License Agreement. Full details of this license are available from the Intel Sample Source Code License Agreement web page [1].

Publicly available documentation

Documentation on the Javascript interfaces is available from the following URL: http://software.intel.com/en-us/articles/intel-cpu-web-api-documentation-and-examples/

Documentation on the WebAPI implementation is not currently publically available.

References

[1] http://software.intel.com/sites/whatif/webapis/license/intel_webapis_license.html

Reinforcement Learning

Brief Description

The Reinforcement Learning (RL) algorithm for user Quality of Experience (QoE) evaluation is an advanced control loop algorithm developed to estimate the user QoE and to find an optimal policy to optimize it.

Programming artifacts

The Reinforcement Learning (RL) algorithm is based on Q-Learning technique has been developed using Java.

Technologies used

The Java implementation of the RL engine relays on a library called Piqle (De Comité, 2005).

Runtime pre-requisites

To have a JVM.

IPR

Core program, examples and documentation of Piqle library are all available under the GNU Lesser General Public License at Sourceforge (http://sourceforge.net/projects/piqle/).

Publicly available documentation

Piqle Java Library - http://piqle.sourceforge.net/

Network Organizer

Brief Description

The Network Organizer realizes the unification of the access to the content in the home system at the file system level. It was developed piartially for the EU Figaro project.

Programming artifacts

The Network Organizer automates the required operation to mount or export a data folder under different communication protocols. The user wanting to include a folder into the global sharing realized by the Network Organizer contacts it via an http connection (web-like server) and exchange with it the necessary information (in particular login and credentials) to realize the ad-hoc mount/export operation.

Technologies used

The Network Organizer is based on the Unix Virtual File System abstraction layer.

Runtime pre-requisites

The Network Organizer may require ad-hoc VFS-Fuse modules depending on the communications protocls used.

Known software requirements

The Network Organizer runs in an Ubuntu environment (10.04).

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

No documentation is yet publicly available

ZigBee GAL

Brief Description

The ZigBee GAL (Gateway Abstraction Layer) is a C++ software which implements the specifications of the ZigBee Gateway and provides a proper abstraction of ZigBee devices and ZigBee networks that enables fast time-to-market via a high-level software API. Through the GAL, an external applications can control or obtain data from individual ZigBee nodes. The ZigBee Home Automation profile (HA 1.0) is supported.

Programming artifacts

Implemented RPC (Remote Procedure Calls):

- REST
- Java interface for OSGi

Technologies used

- C++ GAL for Linux
- Java for Linux
- OSGi
- REST

Runtime pre-requisites

Linux with Java SE runtime OSGi Execution Environment

Known software requirements

The Java headless runtime (jre 1.6.0_10 di SUN) runs on the Linux 2.6 kernel The OSGi Executione Environment (OSGi: Equinox 3.5.2) runs on Java

IPR

All GAL software is copyright of Telecom Italia and will be released under a FRAND (Fair Reasonable and Non Discriminatory) License

Publicly available documentation

ZigBee Alliance - ZigBee Gateway white paper (http://www.zigbee.org/imwp/download.asp?ContentID=16883) ZigBee Alliance - ZigBee Gateway spec. ZigBee Alliance - Home Automation Profile spec. HGI (Home Gateway Initiative)- "HGI-RD008-R3 - HG Requirements for Software Execution Environment" Documentation on the OSGi application that interacts with the GAL and provide the user with a web interface for Network discovery, Device discovery, Communication with a first set of HA devices, is not currently publically available.

NOX based OpenFlow controller

Brief Description

In an Open Networking-based network architecture, open network devices (switches, routers, etc.) are controller by a centralized entity called ‘controller’. The OpenFlow protocol defines the message exchange between the controller and the OpenFlow nodes of the network.

NOX is a Network Operative System on which an OpenFlow controller can be built. On the network side, NOX interacts directly with open network devices by means of OpenFlow instructions. On the application side, NOX exposes an high level API in order to allow network applications to exchange OpenFlow messages with the network.

IPR

All NOX based controller source and documentation is released under version three of the GNU General Public License (GPLv3).

Publicly available documentation

OpenFlow: http://www.openflow.org/

The OpenFlow white paper: http://www.openflow.org/documents/openflow-wp-latest.pdf

NOX: http://noxrepo.org/wp/

NOX white paper: http://noxrepo.org/doc/nox-ccr-final.pdf

Network Element Virtualizer

Brief Description

In a cloud based network scenario, resource abstraction, aggregation and partitioning represents the basic operations defined on the virtual network resources. The Network Element Virtualizer represents the glue technology between physical resources (e.g. network interfaces and links) and the cloud infrastructure. It implements monitoring and configuration features interacting with the network element, via specific modules, including a TL1 client. It provides a standard management interface of the Network Element, translating the incoming, outgoing and autonomous messages of the TL1 protocol into the internal messages manageable at the cloud infrastructure level.

Programming artifacts

Network Element Virtualizer provides the building blocks for monitoring and configuring a virtual transport network, abstracting, partitioning and aggregating physical network element resources. The Network Element Virtualizer implements a Physical Resource Adapter, a software layer that transforms instances of virtual resources in physical resources and vice-versa. The mechanisms used in the interaction between the NetIC and each specific physical resource, depends on the interface offered by each physical device. In order to manage the heterogeneity of the resources, NEV implements the adapter to device-specific management interfaces. Each adapter must provide translation functionalities for the subset of the abstract messages that is supported by the specific physical device.

The Network Element Virtualizer and related software components are implemented both in Java and C++.

Technologies used

The Network Element Virtualizer can be extended developing new components in Java and C++.

Runtime pre-requisites

The Network Element Virtualizer has been built and tested on Ubuntu.

Known software requirements

N/A

IPR

Network Element Virtualizer is a software relized by Alcatel-Lucent. It is free for all FiWare partners, for all the duration of the FiWare project.

Publicly available documentation

Physical Resource Adapter: http://www.geysers.eu

TL1 protocol: Transaction Language 1 - http://telecom-info.telcordia.com/site-cgi/ido/docs.cgi?ID=SEARCH&
DOCUMENT=GR-831&

DRAGON

Brief Description

DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) is a GMPLS control plane that enable dynamic provisioning of network resources on an inter-domain basis, across heterogeneous network technologies.

IPR

This product is licensed under General Public License (GPL) [1]

Publicly available documentation

DRAGON: Dynamic Resource Allocation GMPLS via GMPLS Optical Networks http://dragon.maxgigapop.net/
twiki/bin/view/DRAGON/WebHome

References

[1] http://dragon.east.isi.edu/twiki/bin/view/DRAGON/WebHome

OpenEPC Platform

Brief Description

The Fraunhofer FOKUS OpenEPC is a software implementation inspired by the innovative all-IP aspects of the 3GPP Evolved Packet Core (EPC).

OpenEPC enables prototyping of IP connectivity related features like QoS and charging, mobility management, access and security for 3GPP and non-3GPP wireless technologies including 3GPP Long Term Evolution (LTE). It supports the new paradigms such as Machine Type Communication, Mobile Clouds, and IP Multimedia Subsystem (IMS). OpenEPC toolkit is currently at Rel. 2 further extending the core network functionality based on and updated to 3GPP Release 9 and 10.

Features

The OpenEPC toolkit offers a self-contained environment running on of-the-shelf hardware for starting testbed setups of Evolved Packet Core systems. The testbeds enable developers to prototype, to measure, to test or to simply perform research for future mobile applications and for optimizing core network functionality.

* **Core Network Mobility Management: Basis Functionality** - includes the transparent vertical handover using Proxy Mobile IP (PMIP) protocol and the correspondent functionality in the evolved Packet Data Gateway (ePDG), S-GW and PDN GW. More information available at http://www.openepc.net/components/mobility_management.

* **Policy and Charging Control** - OpenEPC Rel. 2 includes the policy control mechanisms through the Policy and Charging Rules Function (PCRF). Its policy decisions are based on application requirements (Rx interface) and subscription profile (proprietary Sp implementation) and are transmitted and installed to the Policy and Charging Enforcement Function (PCEF) located in the PDN GW and to the Bearer Binding and Event Rules Function (BBERF) located in the access network specific gateways. New releases may provide an Offline Charging System prototype through a Charging Data Function (CDF) and a Charging Gateway Function (CGF) and a Billing System stub. Current features description available at www.openepc.net/components/policy_charging_control/

* **Subscription Management and AAA** - OpenEPC enables subscriber-based AAA based on an extended Home Subscriber Server (HSS) interacting with an SGSN prototype, MME and a 3GPP AAA Server. More description on the features is available at http://www.openepc.net/components/subscription_management/index.html

* **Client Mobility Management** - OpenEPC supports policy-based access network discovery and operator assisted access selection through an Access Network Discovery and Selection Function (ANDSF) in the core network and a client mobility manager for Linux OS based mobile devices. Information available at http://www.openepc.net/components/client_mobility_support/index.html

* **Core Network for LTE** - OpenEPC features updated LTE core network including MME and Serving GW functionality. Additionally an eNodeB emulator is offered for testbeds without radio license. Information available at http://www.openepc.net/components/Core_Network_for_LTE/index.html

* **Connection to access networks** - OpenEPC conects either directly or through third party radio hardware and software to various access networks including LTE, UMTS, GPRS, WiFi, WiMAX, CDMA 2000 etc. More information available at http://www.openepc.net/project_info/connection_access/index.html

* **Connection to applications**. OpenEPC includes a Diameter interface towards the different application platforms for resource reservation and data path event notification as well as a Diameter interface towards the Home Subscriber Server for shared subscription profile information between core networks and applications. Data path communication is plain IPv4 or IPv6. Information on a set of example demonstration enablers is available at http://www.openepc.net/components/epc_demonstration_enablers/index.html

Technologies used

The OpenEPC was developed mainly in C programming language as GPL-free platform using a highly modular structure named Wharf. More information available at http:/ / www. openepc. net/ components/ testbed_prototyping_experiments/index.html

Runtime pre-requisites

OpenEPC platform was built on off-the-shelf x86 platform using the Ubuntu OS. More information at http://www. openepc.net/project_info/hardware_requirements/index.html

Known software requirements

1. libdrizzle - Drizzle Client & Protocol Library
2. libgcrypt - General purpose cryptographic library
3. libxml2 - The XML C parser and toolkit of Gnome
4. PHP5 - General-purpose scripting language suited for Web development

The OpenEPC software is to be used with the following components, which are not part of OpenEPC toolkit:

1. MySQL - Open Source database software
2. ISC-DHCP - Open Source DHCP implementation
3. Apache2 - Open Source HTTP Server
4. Squid - Cache proxy for Web supporting HTTP, HTTPS, FTP and more
5. Sofia-SIP - Open Source SIP User-Agent library
6. GStreamer - Library for constructing graphs of media-handling components

IPR

This product will be licensed under FRAND (Fair Reasonable and Non-Discriminatory [2]) Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

The official page: http://www.openepc.net/index.html

Device Connection Platform

Brief Description

The Device Connection Platform is an Ericsson product that provides a cost and business optimized service to networks dealing with M2M communication.

Programming artifacts

The Device Connection Platform provides the needed interfaces for easy two way initiated communication and addressing between 3GPP radio devices and enterprise applications. DCP implements standard VPN connectivity through a GGSN as well as Device Access Enabler (DAE) connectivity enabling devices to be addresses with a fully qualified hostname over internet towards the enterprise applications, communication with the devices is done over the GRX (GPRS Roaming Exchange) network. The software components in DCP is mainly implemented in C/C++ and Java.

Technologies used

- Linux
- MySQL
- C/C++
- Java
- L2/L3 switches routers
- Firewalls
- OpenSAF

Network

Use the Ericsson Device Connection Platform providing device accessibility. Providing

- A true two way initiated connectivity, meaning a connection can be initiated from either the Enterprise or from the Device or both.
  - A device can act as a server, client or both just as the Enterprise can be a client or server
- All devices can be addressed using a unique DNS registered device hostname that initially will be based on the IMSI number
  - Further support will be added to allow for Enterprise controlled alias names for their devices
- Device Access Control: With the device access enabler the Enterprise can decide exactly who and from where a device can be accessed.
- Internet Access Control: Equally important is that it will be possible to control what the devices can access on the Internet.
- Secure communication: All communication between the Enterprise and the Device Access Enabler can be done securely and with confidentiality over the Internet.
  - Between the Device Access Enabler and the Device all communication is done over controlled and secure networks
  - Meaning that communication can be secured even towards devices that do not support secure protocols.

The philosophy and design behind the Device Access Enabler has been that it should be completely protocol agnostic and thereby allow an Enterprise application to communicate with any type of device, new or old, as long as it can talk TCP/UDP over IP.

- How the devices can be reached has been addressed as well as the possibility to initiate any type of IP connection from either the device or from the enterprise application.
- Thus allowing for direct proprietary Enterprise – Device IP communication as well as standardized communication provided by some higher level service enablement layer

The focus for the Ericsson DCP has so far been towards radio (3GPP) connected devices but is foreseen to include devices connected using additional methods, such as for example ADSL, in future releases.

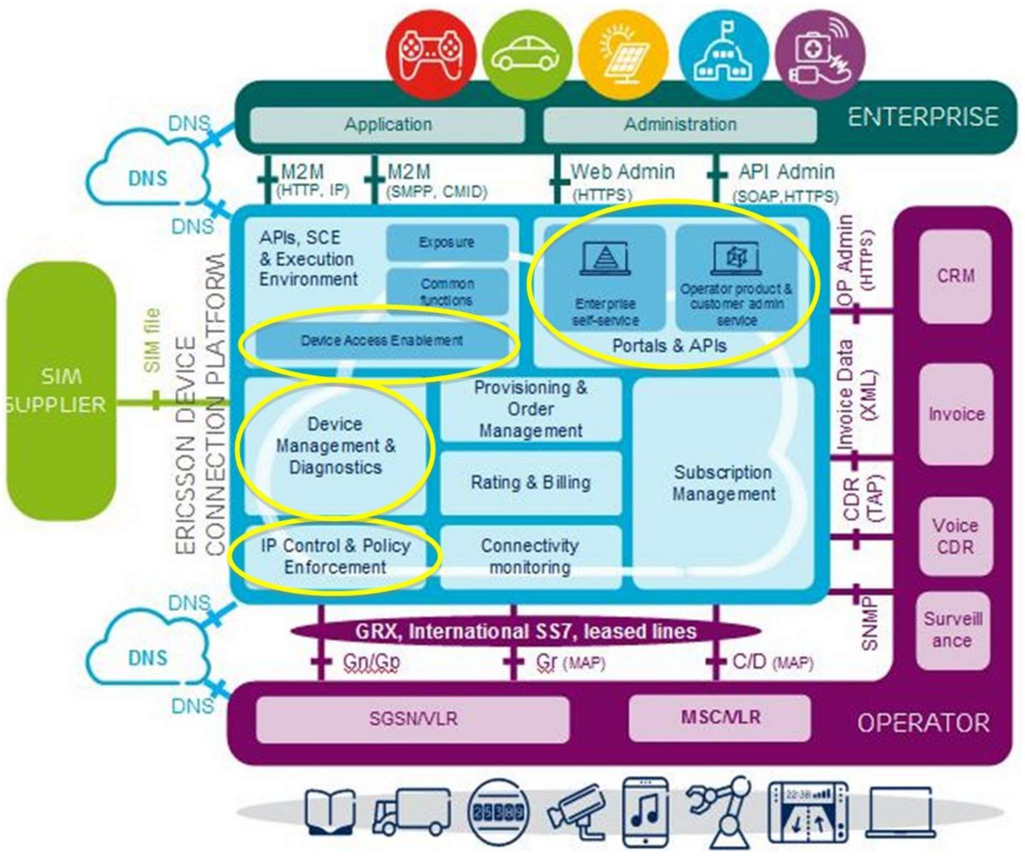
Likewise it is also foreseen that the platform will/must in the future include support for QoS both in the interface to/from the devices, including the radio/air interface, as well as to/from Internet.

Remote Management

The Ericsson Device Connection platform contains API's for

- Reporting of statistics of connection quality and usage
- Remote device management, such as
  - Reading local device statistics/configuration
  - Firmware upgrade/downgrade
  - Remote wake-up in case device is asleep (no PDP-Context)

Functional Blocks



Device Access Enablement	Enables transparent and secure access between the Enterprise and radio connected devices over Internet.
IP Control and Policy Enforcement	Also enables enforcement of traffic limits and QoS parameters to group of devices.
Portal & APIs	Enables self administration of devices and subscriptions from the Enterprise either through a WEB portal or through APIs.
Device Management & Diagnostics	A function that provides remote management capabilities of the devices for the Enterprise, such as: • Reading local configuration data • Firmware upgrade • Wake-up

Runtime pre-requisites

N/A

Known software requirements

N/A

IPR

This product has IPR associated to it.

Publicly available documentation

http://www.ericsson.com/ourportfolio/products/device-connection-platform?nav=fgb_101_973

API mediation

Brief Description

The API mediation component provides a set of APIs to several actors interacting with the network infrastructure, and includes the functionalities needed to manage the exposed interfaces. The API mediation component deals with publishing, discovering and exposing APIs to stakeholders as well as managing some non functional aspects such as provisioning and monitoring. It doesn't create or host APIs nor manage/provision customers, but makes it possible for API providers to register new APIs sets to the Exposition, QoS and resource management.
This is a server not a software

Programming Artifacts

This is a layer between servers, services, administration and users. To servers it acts as a client proxy, to the client it acts as a server proxy, to services it exposes an administration interface. To the operator it exposes also an administration interface.

Technologies used

J2EE

Runtime pre-requisites

This asset has no runtime pre-requisites’

Known software requirements

A public available kit is necessary is some cases. It's needed in cases where the developer is not so skilled and need basic examples and libraries of useful tools. It's done on an ad hoc basis.

IPR

Proprietary server available under FRAND terms

Publicly available documentation

Work in progress, This asset is not publicly available

Fast Telecom Service helper

Brief Description

The Telecom AS APIs Provides functionalities that make it possible to create a call, terminate a call, bridge calls.
Currently as a platform the Telecom enabler it offers "audio-mixing for conferencing, automated dialog (speech synthesis, voice recognition, DTMF recognition). The north interface uses SOAP (order from service logic, service notifications to logic) and the south interface works with SIP / RTP. SOAP REST is possible but not yet in place directly on this AS.
It is a server, not a software.

Programming Artifacts

This is a layer between servers, services, administration and users. To servers it acts as a telco client proxy, to the client it acts as a Telco server proxy, to services it exposes an administration interface making it possible to create new services, modify and destroy them easily. To the operator it exposes also an administration interface.

Technologies used

Currently as a platform the Telecom enabler it offers "audio-mixing for conferencing, automated dialog (speech synthesis, voice recognition, DTMF recognition). The north interface uses SOAP (order from service logic, service notifications to logic) and the south interface works with SIP / RTP. SOAP REST is possible but not yet in place directly on this AS.

Interfaces functionalities: VoIP signaling and media to telecom networks, orders / notifications to the service logic, CDR (not existing on this AS, made by the layer IAPI) for the accounting, service configuration via an OAM. The multi-service aspect will involve notions of limitations of a priori traffic service (not yet available on this AS immediately).

Runtime pre-requisites

No runtime

Known software requirements

This asset has no runtime pre-requisites

IPR

Proprietary server available under FRAND terms

Publicly available documentation

Work in progress, This asset is not publicly available

Voicemail enabler

Brief Description

The Voice&Videomail enabler is a server that offers the possibility to developers, to manage voicemail or videomail contents and settings facilities such as voicemails or videomail provisioning, voicemail or videomail retrievals, deposit of messages, greetings managements,text and other notifications, Visual Voicemail rendering, getting transcription of voicemail to text.
It is a server, not a software.

Programming Artifacts

This is an enabler, it is basically a server. The client part could be a phone client or a Web browser.

Technologies used

The architecture of the platform implements components such as:

- HTTP rest API
- VVM OMTP API with
- IMAP Proxy .
- Telephony and Video User Interface through SIP and/or PSTN
- Monitoring and diagnostic systems

Runtime pre-requisites

This asset has no runtime pre-requisites'

Known software requirements

Client must be VoiceXML 2.0 compliant. This is no software requirements here, only compliance to the specification.

IPR

Proprietary server available under FRAND terms

Publicly available documentation

Work in progress, This asset is not publicly available

Intelligent Network Identity Management

Brief Description

Network ID management is an asset enabling forward of incoming requests to a user to a specific user terminal based on system-default or user-modified preferences. In this way the originator of the session must not care about certain and exact user ID (like landline/office/mobile number), it just uses a generic ID of another party to acheive it. During session setup available resources of registered devices are analysed in addition to the preferences to enable session on the most suited terminal of the user.

Programming artifacts

IMS is prerequired for the usage. Thereby different implementation variants can be performed in order to deploy this asset in the network using IMS in the control plane. The SIP INVITE message is received in the network and analysed in order to decide to which identity/-ies of the user the request must be forwarded. The classification can be performed based on source identity of the message, type of service, requried QoS, QoS available for particular devices of called subscriber, etc.

Technologies used

IP Multimedia Subsystem must be deployed in the network for the asset. It can be implemented either as a separate application server (AS) or by issuing SIP re-direct message. Then either AS or S-CSCF applies rules for classification of the received request and forwards the request to the appropriate device of the called subscriber.

Runtime pre-requisites

IP Multimedia Subsystem. Functions of the OpenIMS from Fraunhofer Fokus platform have been extended to support this asset (P-CSCF, S-CSCF).

- Linux based implementation.
- Additional functions are based on SIP stacks osip and eXosip.

IPR

This asset will be licensed under FRAND Terms according to pre-requisites of the FI-PPP program.

Publicly available documentation

The documentation is not publicly available

Network-centric Resource Management

Brief Description

Resource Management is an asset enabling analysis of available resources for particular user devices, reservation and monitoring of required resources for an active user session and feedback opportunities in the case required resources may not more be provided for the subscriber.

Programming artifacts

Extended RACS and NASS as specified by ETSI TISPAN. Both systems are extended to gather additional information specific for multi interface terminals and to synchronously analyse resources available for the user by using any of its network interfaces. RACS thereby is used for the selection of interface to be initially, taking handover decision in the case of insufficient resources with the serving AN due to congestion or bad link quality. NASS is used to reconfigure delivery of IP flows to and from terminal device on flow basis.

Technologies used

Resource and Admission Control Subsystem (RACS) and Network Attachment Subsystem (NASS) are utilised in the available implementation. ETSI TISPAN Diameter-based Gq interface is used to communicate QoS requirements from Application Function to the transport plane that is also extended to provide feedback to the application about modifying resources available for the user in the transport network. In order to assess available resources in the network additional function must be able to monitor resources for particular subscribers. Analysing of e.g. established bearers in 3GPP access networks is insufficient for this purpose while exact transport parameters and available resource blocks must be analysed. These functions are challenging especially for 3GPP accesses since scheduling information is not available via a specified interface. Additional resource monitoring functions have been implemented for following access techniques: - IEEE 802.11 DCF - IEEE 802.11 based mesh networks with OLSR - IEEE 802.16 - 3GPP LTE (here based on Fraunhofer HHI equipment)

Runtime pre-requisites

Linux based implementation. Java based Diameter stack, PHP5, java based monitoring of available resources for different access technologies.

IPR

EU patent pending:

- "A System and a method for providing improved quality of a communication service"
- No. 08156992.3 [1] (27.05.08)

This asset will be licensed under FRAND Terms according to pre-requisites of the FI-PPP program

Publicly available documentation

Not available now.

References

[1] https://data.epo.org/publication-server/getpdf.jsp?pn=2129061&ki=A1&cc=EP

Seamless Network Connectivity

Brief Description

Seamless Network Connectivity enables QoS supported very low-delay mobility between different accesses technologies especially within Layer 2 connected networks. While other solutions deals with packet buffering that is not always actual for multimedia communications, this asset focuses on extremely low delay handovers (<5ms) whereby packet losses are very rare. Additional techniques are in planning to avoid them completely. The focus is on load distribution generated by multiple IP flows of the user among multiple interfaces available on the terminal. Traditional horizontal handovers are also supported while inherent mechanisms of particular access technologies can also be used for this purpose.

Programming artifacts

NASS specified by ETSI TISPAN is extended to gather additional information about multiple interfaces of mobile terminal. This information is required to uniquely identify the terminal with its multiple network interfaces in the network in order to determine which access nodes can be used for the communication with the terminal. Via e4 interface this information is shared to the RACS that is then able to determine available interfaces for specific terminal on each AN that can be used by the terminal. Seamless Network Connectivity supports QoS enabled communication via multiple interfaces due to tight interaction with Resource Management enabler. Resource Manager knows about QoS demands of all sessions maintained by the user while Seamless Network Connectivity Enabler knows about all accesses available for terminal. Scheduling of IP flows generated by the user can then be performed by selecting the most suited access for every IP flow of the user.

Technologies used

DHCP protocol has been extended to transfer additional information from terminals to the network. Also reconfiguration of IP addresses used on different interfaces of the terminal for specific IP flows is signalled to the terminal from the network with help of DHCP.

Runtime pre-requisites

- DHCP configured devices.
- L2 network connecting access networks.

The client software is implemented for Windows XP based devices.

IPR

EU/US patents owned by Telekom:

- "A new flow based Layer 2 handover mechanism for mobile node with multi network interfaces"
- EU: No. 1988679 [1] (17.03.10)
- US: 12/104,803 [2] (Nov. 2011)

This asset will be licensed under FRAND Terms according to pre-requisites of the FI-PPP program

Publicly available documentation

Patent texts.

References

[1] https://data.epo.org/publication-server/pdf-document?PN=EP1988679%20EP%201988679&iDocId=7287201&iepatch=.pdf

[2] http://www.google.com/patents/download/12_104_803_FLOW_BASED_LAYER_2_HANDOVER_M.pdf?id=e8OvAAAAEBAJ&output=pdf&sig=ACfU3U0GIY-ZUlNlp_y5wBFUtxK8mKnFgw&source=gbs_overview_r&cad=0