Deliverable D200.1

FIspace Design and Release Plan

WP200

Project Acronym & Number:	FIspace – 604 123
Project Title:	FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics
Funding Scheme:	Collaborative Project - Large-scale Integrated Project (IP)
Date of latest version of Annex 1:	03.10.2013
Start date of the project:	01.04.2013
Duration:	24
Project Website:	http://www.FIspace.eu
Document Identifier:	FIspace_D200.1-final2.docx
Date:	12.07.2013
Revision:	008
Editor:	Michael Stollberg (SAP)
Contributors: (ordered by project partner)	• **ATB**: Gunnar Große Hovest, Philip Reimer • **UDE**: Andreas Metzger, Clarissa Marquezan • **SAP**: Rene Fleischhauer, Benjamin Heilbrunn, Steffen Buzin, Michael Stollberg • **IBM**: Eliezer Dekel, Yael Dubinsky, Gidon Gershinsky, Fabiana Fournier, Sarit Arcushin • **ATOS**: Carlos Maestre Terol, Elies Prunes Soler • **KOC**: Serdar Arslan, Seyhun Futaci, Özgür Sönmezer • **TOG**: Scott Hansen • **AST**: Monika Solanki • **NKUA**: Aggelos Groumas, Lampros Katsikas, Sokratis

	Barmpounakis • **UPM**: Tomás Robles, Ramón Alcarria • **LimeTri**: Timon Veenstra
Internal Quality Reviewers:	• Rod Franklin (KN) • Adrie Beulens (WU) • Timon Veenstra (LimeTri)

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change History

Version	Notes	Date
001	Creation of document & initial structure	13.05.2013
002	First Inputs integrated (esp. Sec. 3)	14.06.2013
003	Revision of Structure & Content Plan (esp. for Release Plans and Component Design in Section 3)	21.06.2013
004	Added Introduction and Section 2	29.06.2013
005	Integrated revised sections on components in Sec 3	02.07.2013
006	Integrated Release Plan & Methodology Part, moved to Sec 3	08.07.2013
007	Added revised component design sections (now in Sec 4)	09.07.2013
008	Pre-final revision and preparation for internal quality review	12.07.2013

The FIspace Project

Leveraging on outcomes of two complementary Phase 1 use case projects (FIspace & SmartAgriFood), aim of FIspace is to pioneer towards fundamental changes on how collaborative business networks will work in future. FIspace will develop a multi-domain Business Collaboration Space (short: FIspace) that employs FI technologies for enabling seamless collaboration in open, cross-organizational business networks, establish eight working Experimentation Sites in Europe where Pilot Applications are tested in Early Trials for Agri-Food, Transport & Logistics and prepare for industrial uptake by engaging with players & associations from relevant industry sectors and IT industry.

Project Summary

As a use case project in Phase 2 of the FI PPP, FIspace aims at developing and validating novel Future-Internet-enabled solutions to address the pressing challenges arising in collaborative business networks, focussing on use cases from the Agri-Food, Transport and Logistics industries. FIspace will focus on exploiting, incorporating and validating the Generic Enablers provided by the FI PPP Core Platform with the aim of realising an extensible collaboration service for business networks together with a set of innovative test applications that allow for radical improvements in how networked businesses can work in the future. Those solutions will be demonstrated and tested through early trials on experimentation sites across Europe. The project results will be open to the FI PPP program and the general public, and the pro-active engagement of larger user communities and external solution providers will foster innovation and industrial uptake planned for Phase 3 of the FI PPP.

Project Consortium

- DLO; Netherlands
- ATB Bremen; Germany
- IBM; Israel
- KocSistem; Turkey
- Aston University; United Kingdom
- ENoLL; Belgium
- KTBL; Germany
- NKUA; Greece
- Wageningen University; Netherlands
- PlusFresc; Spain
- FloriCode; Netherlands
- Kverneland; Netherlands
- North Sea Container Line; Norway
- LimeTri; Netherlands
- Kühne + Nagel; Switzerland
- University Duisburg Essen; Germany
- ATOS; Spain
- The Open Group; United Kingdom
- CentMa; Germany
- iMinds; Belgium
- Marintek; Norway
- University Politecnica Madrid; Spain
- Arcelik; Turkey
- EuroPoolSystem; Germany
- GS1 Germany; Germany
- Mieloo & Alexander; Netherlands
- OPEKEPE; Greece
- Innovators; Greece

More Information

Dr. Sjaak Wolfert (coordinator)
LEI Wageningen UR
P.O. Box 35
6700 AA Wageningen

e-mail: sjaak.wolfert@wur.nl
phone: +31 317 485 939
mobile: +31 624 135 790
www.FIspace.eu

Document Summary

This report presents the first deliverable of work package WP200 of the FIspace project that is concerned with the development of the FIspace, more precisely the technical specification and implementation of generic software infrastructure that shall enable the envisioned seamless collaboration in business networks and support the development, provisioning, and consumption of FIspace Apps which shall provide novel, value-added functionalities for business collaboration in several application domains. In particular, this deliverable presents the initial overall and consolidated technical design of the FIspace and its main technical building blocks along with the initial development and release plan, therewith providing the initial results of tasks T210 – T280 as defined in the Description of Work (DoW). It is important to note that – following the agile methodology applied in the project – the initial technical design and work plans presented here will be refined and revised where necessary throughout the upcoming project milestones.

The FIspace is a value-added Collaboration Space, designed as a Cloud-based platform and applying various Future Internet technologies provided as FI PPP Generic Enablers, that shall enable actors operating in Collaborative Business Networks (e.g. businesses, authorities, public & private service providers) to conduct cross-organizational collaboration, communication, and coordination of activities in a seamless, efficient, and quick adoptable manner, therewith overcoming pressing challenges arising in various industries (e.g. Agri-Food, Transport and Logistics, as well as other domains). In the FIspace project, WP200 is concerned with the development of the generic software infrastructure for this. As the first deliverable, this report presents the initial technical design and release plan of the FIspace. In particular, it explains the overall conceptual design of the FIspace, depicts the methodology and work plan for the technical specification and development planned in upcoming milestones, and – as the main part – presents the consolidated conceptual design of the main components of the FIspace, including the initial release plans and validation plans for various Generic Enablers provided by FIWARE:

(1) the *‘Front-End’* as the main point of access to all FIspace features and apps

(2) the *‘FIspace Store’* that supports the provisioning, consumption, and accounting of FIspace Apps

(3) the *‘Real-time B2B Collaboration*’ core modules that enable the event-driven provisioning of information on collaborative business activities to all involved actors at the right time

(4) the *‘System & Data Integration’* support for connecting external business, legacy, and 3rd-party systems as well as IoT-enabled systems to the FIspace

(5) the *‘Operating Environment’* that ensures the technical interoperability of the FIspace components and its reliable operation

(6) the *‘Security, Privacy, and Trust’* framework that will ensure the FIspace to be secure by design,

(7) the *‘Development Environment’* for FIspace App Developers and Business IT Engineers who configure and customize the FIspace for individual business needs.

The report is structured as follows:

- Section 1 introduces into the report, including an overview of the agile methodology that is applied throughout the project and a detailed outline of the deliverable,
- Section 2 provides a comprehensive overview of the overall conceptual design of the FIspace, including the a high-level architecture identifying the main components, the illustration of the overall operational model at hand of a sample scenario from the trials, and the definition of the principle user roles with the main usage processes,
- Section 3 presents the initial version of the consolidated development and release plan for the FIspace development in WP200, and introduces the agile methodology that is applied throughout the project,
- Section 4 encompasses the detailed initial technical design of the seven main components of the FIspace as outlined above, and
- Finally, Section 5 summarizes the report and outlines the plan for the upcoming milestones.

Abbreviations & Acronyms

Acronym	Explanation
Agile	Here referring to *agile software development* as methods for iterative and incremental development where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.
App	Short form for an application running in mobile devices
ASP	Application Service Provisioning (traditional model of software delivery)
B2B	Business to Business
B2C	Business to Consumer
Back-End	Layer of the FIspace Platform for facilitating the integration of external systems (legacy & standard business systems, 3rd-party services...)
BCM	Business Collaboration Module
BPEL	Business Process Execution Language (OASIS standard)
Cloud	Infrastructure for providing computational resources (servers, virtual machines, etc.) in a centrally management environment; platforms, services, and applications can be deployed on this in order to minimize the TCO (total cost of ownership) as well as other issues relevant for enabling cheap and quick development of high-quality solutions
DOM	‘Data Object Model’ (technical term): the technical model of a data object describing its basic structure and data types
DoW	Description of Work (Annex I of the Grant Agreement describing objectives, organization, and the detailed work plan of the project)
Ecosystem	An economic system comprised of various stakeholders that conduct business together, forming an overall value added network
EPM	Event Processing Module
ERP	‘Enterprise Resource Planning’ system: overall term for standard business systems for managing the resources of an enterprise (e.g. customers, employees, sales & orders, etc.); often used as synonym for the older solutions from SAP (e.g. SAP R3 systems, SAP Business Suite)
FI PPP	Future Internet Private-Public Partnership
FI-WARE	FI PPP project that develops the ‘Future Internet Core Platform’, which consists of so-called ‘Generic Enablers’ that shall be used for realizing the FIspace Platform; see public website: www.fi-ware.eu
GDL	‘Gadget Description Language’: technical annotation language for gadgets (self-contained technical elements that can be added & configured to web-based UIs with respect to individual needs)
GE	Abbreviation for ‘Generic Enabler’; term used in the context of the FI PPP, referring to one of the specific generic technologies that are developed in the course of the FIWARE project
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
IDM	Identity Management - general term for authentication, authorization ,roles, and privileges/permissions within or across system
Interface	Technical element for (automated) information exchange between components
IoS	‘Internet of Services’: any kind of service (technical, business, non-technical) that is accessible over the Web and can be re-used in various application scenarios, fostering the idea of service-orientation
IoT	‘Internet of Things’: technologies for receiving information about real-world objects (e.g. via sensor networks) and enable their treatment as programmatically accessible entities in software environments
IPS	Intrusion Prevention System (system for recognition / prevention of attacks on computer systems)
LDAP	Lightweight Directory Access Protocol, protocol for accessing and maintaining information in distributed directories
Linked USDL	The representation / refined definition of the ‘Unified Service Description Language’ (see USDL) based on the Linked Data approach; this is the basic service description model supported by FIWARE
OMG	Object Management Group, technology standardization body
PaaS	Platform as a Service
PCS	Port Community System
PKI	Public Key Infrastructure (security technology)
REST Service	A Web Service accessible as a Web Resource, applying to the principles of Representational state transfer (REST)
RESTful	Technical design principle for realizing access to web-based services and resources; also see REST Service
RFID	“Radio-frequency identification”: a IoT technology that uses radio waves to transfer data from an electronic tag

	(RDIF tag) attached to a real-world object into a software environment via specialized readers
SaaS	Software-as-a-Service, modern delivery model for software systems
SLA	Service Level Agreement
SOAP	Protocol for exchanging structured information, most commonly used in traditional Web Services
SSL	Secure Sockets Layer: network protocol for secure information transfer
SSL	Secure Socket Layer – security technology for reliable information exchange over the Web
SSO	Single Sign on – access control model for a user to various systems
SVG	Scalable Vector Graphic
TEP	Transport Execution Plan (detailed description of a part of logistics process)
TPM	Transport Planning Module
UI	User Interface, usually referring to graphical user interfaces for human-machine interaction
UML	Unified Modelling Language: established standard for model-driven software engineering published by OMG
USDL	‘Unified Service Description Language’: comprehensive description model for both technical and business services; in standardization process of W3C as an Incubator (see http://www.w3.org/2005/Incubator/usdl/)
VPN	Virtual Private Network (secure & reliable connections for information transfer over the Web)
VPN	Virtual Private Network – security technology for reliable information exchange over the Web
W3C	World Wide Web Consortium, standardization body for web technologies (see www.w3c.org)
Widget	A part of a GUI that can be added to a web-based Front-End
WSDL	Web Service Description Language (W3C Recommendation)

Contents

List of Figures

List of Tables

1 Introduction

The overall aim of the FIspace project is to implement a novel Future Internet enabled cloud-based platform for enabling easy, effective, and seamless collaboration in business networks across organizational borders – called *FIspace* – which is validated in 8 trials from the Agri-Food, Transport, and Logistics domains and prepared for large-scale experimentation and industrial uptake in Phase 3 of the FI PPP and beyond. Work package WP200 is concerned with the development of the FIspace, more precisely: the technical specification and implementation of generic software infrastructure that shall enable the envisioned seamless collaboration in business networks and support the development, provisioning, and consumption of FIspace Apps that provide novel, value-added functionalities for business collaboration in several application domains and shall be developed for the 8 project trails as well as by external parties.

As the first deliverable of WP200, this report presents the initial technical design and release plan of the FIspace. It explains the overall conceptual design of the FIspace, depicts the methodology and work plan for the technical specification and development planned in upcoming milestones, and, in particular, presents the consolidated conceptual design of the following main components of the FIspace, including the initial release plan and the validation plans for various Generic Enablers provided by FIWARE[1]: (1) the *‘Front-End’* as the main point of access to all FIspace features and apps, (2) the *‘FIspace Store’* that supports the provisioning, consumption, and accounting of FIspace Apps, (3) the *‘Real-time B2B Collaboration*’ core modules that enable the event-driven provisioning of information on collaborative business activities to all involved actors at the right time, (4) the *‘System & Data Integration’* support for connecting external business, legacy, and 3rd-party systems as well as IoT-enabled systems to the FIspace), (5) the *‘Operating Environment’* that ensures the technical interoperability of the FIspace components and its reliable operation, (6) the *‘Security, Privacy, and Trust’* framework that will ensure the FIspace to be secure by design, and (7) the *‘Development Environment’* for FIspace App Developers and Business IT Engineers who configure and customize the FIspace for individual business needs. Therewith, the report provides the initial results of tasks T210 – T280 as defined in the Description of Work (DoW); please note that – following the agile methodology applied throughout the project – the initial technical design and work plans presented here will be refined and revised where necessary throughout the upcoming milestones of the project.

The remainder of this report is structured as follows:

Section 2 provides a comprehensive introduction and overview of the overall design of the FIspace, including:

- The overall concept with the business relevance and a high-level conceptual architecture, depicting the major building blocks and components of the FIspace solution (Section 2.1)
- An explanation of the overall operational model at hand of an illustrative example (Section 2.2)
- The overall usage model and processes of the FIspace (Section 2.3).

Section 3 describes the initial development plan for the FIspace in WP200, including:

- The initial release plan for the FIspace components with respect to the overall structure and focus on the three releases planned for WP200, the detailed release plans for the individual components are provided in Section 4, and
- The overall agile methodology that is applied throughout the project, with particular attention to the agile development methodology that is established in WP200.

Section 4 provides the initial version of the consolidated conceptual design along with the specific initial release plans for the following main FIspace components, which has been elaborated in several workshops in a scenario-driven manner:

[1] The FIWARE project develops the Core Platform on the Future Internet PPP, in form of so-called Generic Enablers that provide general purpose future internet technologies and re-usable solutions, see www.fi-ware.eu.

- The *FIspace Front-End* that serves as the main access point offering an ‘all you need in 1 place’ user experience with integrated features for seamless business collaboration (Section 4.1)
- The *FIspace Store* that provides the tool-supported infrastructure for providing, finding, and purchasing FIspace Apps (Section 4.2)
- The *B2B Collaboration Core Modules* that consists of the Event Management Engine and the Business Entities Engine which together enable the provisioning of all information to each stakeholder involved in a collaborative business process in real-time (Section 4.3)
- The *System & Data Integration* facilities that enable the quick and easy connection of existing system landscapes (i.e. back-end, legacy, and IoT-enabled systems) to the FIspace (Section 4.4)
- The *Operating Environment* that provides the middleware of the FIspace and ensures the smooth and fault-tolerant interoperability of all technical components (Section 4.5)
- The *Security-Privacy-Trust* framework that ensures secure and reliable handling of information and access control for the FIspace (Section 4.6), and
- The *Development Environment* that provides tool support for developing and instantiating FIspace Apps (Section 4.7).

Finally, Section 5 summaries the report along with an outlook to the upcoming project milestones.

2 FIspace Overall Design

In order to provide a sophisticated basis for comprehending the initial technical design of its components, this section provides a comprehensive overview of the FIspace. For this, the following first explains the overall concept along with the business motivation and relevance and introduces the main technical building blocks (Section 2.1). Then, we outline the planned overall operational model of the FIspace by explaining the concept of FIspace Apps and how this operate on the generic FIspace platform features at hand of an illustrative example (Section 2.2), and finally depict the intended usage model of the FIspace and its main usage processes (Section 2.3).

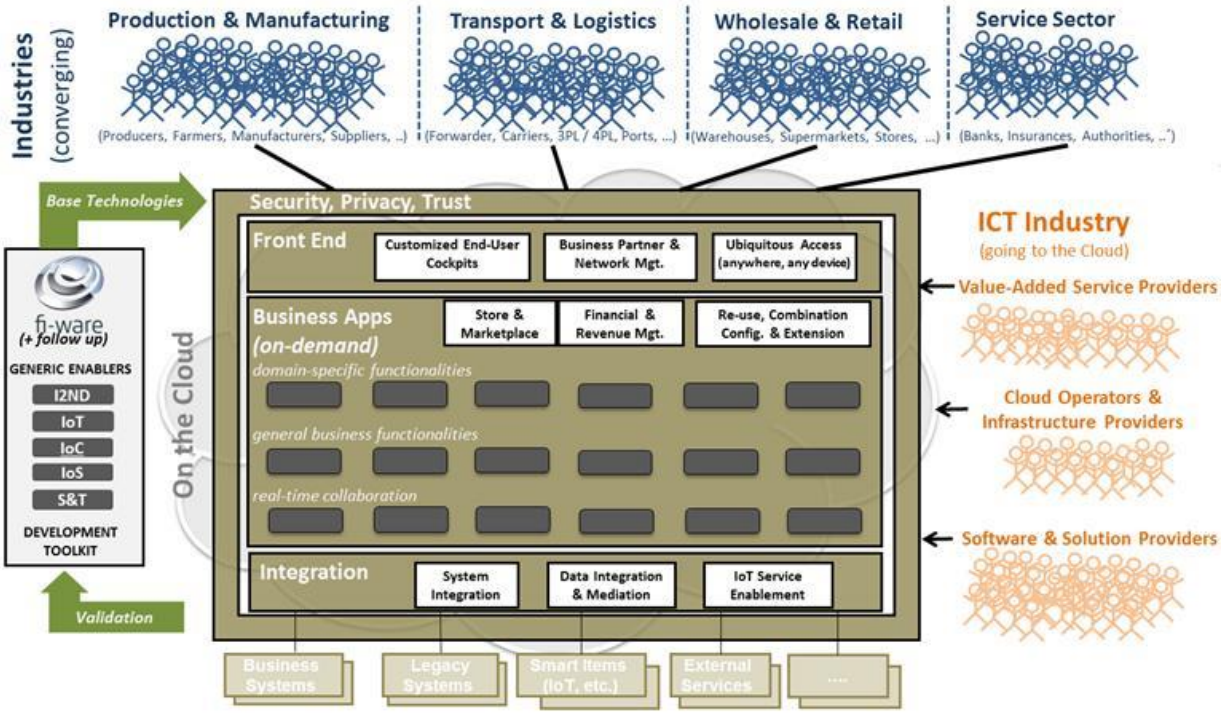
2.1 FIspace Conceptual Architecture

As the starting point for a comprehensive overview, the following first introduces the overall concept of the FIspace by detailing the overall vision with respect to the business relevance in order to satisfy the requirements that are arising across several industries. Then, we provide a high-level conceptual architecture and identify the main technical building buildings of the FIspace whose initial technical design is presented in detail in Section 4.

2.1.1 Overall Concept

The FIspace is a value added Collaboration Space designed as a Cloud-based platform enabling actors operating in Collaborative Business Networks (e.g. businesses, authorities, public & private service providers) in various application domains to find out about one another, determine what services others can provide, and to collaborate on developing and executing solutions to business needs that they might have in a seamless and easy manner. In order to allow for rapid development of high-quality ICT solutions at minimal costs, the FIspace enables collaborators to select, assemble (mash up), and execute functional applications from its Cloud based application store (the FIspace App Store). General business as well as domain-specific functionalities (referred to as ‘Apps’, as the envisioned usage and economic model is similar to mobile apps for smartphones) are developed and placed in the FIspace App Store, through which the Apps can be employed by collaborators to effect business services through the use of the FIspace. New Apps can be developed by re-using features of existing Apps or through the development of completely new Apps using the FIspace App Development Environment. Apps are selected based on a number of criteria including their functionality, pricing model, past reliability, focus, etc. The Apps can be “mashed up” (connected together in an integrated manner) in a rapid and low cost fashion using the mechanisms and tools provided by the FIspace, in a rapid manner at minimal costs. These “mashed up” solutions, possibly composed of multiple Apps, are targeted to address unique business problems, not general problems, and can be “discarded” once the problem or business opportunity has been successfully addressed. The FIspace facilitates such rapid construction and destruction to ensure that businesses can address issues or opportunities in “real time” without having to address the overhead and cost that has plagued the development of traditional monolithic applications.

Figure 1 below depicts the overall vision for the FIspace service. The FIspace is a value added Collaboration Space deployed “in the Cloud” that enables actors operating in Collaborative Business Networks (e.g., enterprises of all sizes, authorities, public and private service providers) in various application domains to seamlessly interact, communicate, and coordinate activities with business partners and to easily create and act in open and dynamic networks of connected businesses – similar to modern Web 2.0 solutions that exist in the B2C world (e.g., Facebook, LinkedIn, Xing, etc.). In addition, the FIspace instantiates a unique business model for enabling the rapid development of high-quality B2B ICT solutions at minimal costs by enabling the provisioning, consumption, and re-use of on-demand solutions in the Cloud. General business, as well as domain-specific, functionalities are developed by IT solution providers (beneficiaries and external providers). These “Apps” are provided via the FIspace App Store, from which the Apps can be consumed and for which new Apps can be developed through the use of the FIspace App Development SDK. App “mash-up” services provided by the FIspace platform allow business collaborators to rapidly create near real time customized solutions that can be executed and managed through the FIspace creating a low cost mechanism for organizations of all sizes to conduct value added business with one another.



BUSINESS BENEFITS	
Industries (enterprises operating in collaborative business networks)	**ICT Industry** (Software & Service Providers)
• Seamless B2B Collaboration (**information exchange, communication, coordination of activities**) • **Rapid & easy development** of customized solutions at minimal costs • **Quick formation & evolution** of open business networks	• Paving the **way to the cloud**, pioneering on both future **technology** and **business models** • Enable new **market & distribution channels** • Facilitate **novel business opportunities**, esp. for market entry & participation for SMEs

Figure 1: FIspace Overall Vision

2.1.2 Business Relevance

2.1.2.1 Motivation

Modern business networks tend to be highly distributed inter-organizational entities spanning country boundaries composed of business partners who have limited insights into the overall network and who are only focused on optimizing their own small part of the value chain. Current ICT services generally support this limited network focus, and thus provide only basic support for inter-organizational data and process integration. This means that complex inter-organizational collaboration activities today must be accomplished through manual efforts.

Technology advancements are also placing increasing strains on existing ICT systems. New technologies for gathering data on field activities, such as new sensor technologies, scanners, and RFID, are creating data collection, distribution and management problems for existing Internet technology. Sharing of these data is also problematic as the requirements for privacy and security of these types of data are poorly supported by existing Internet services.

The lack of robust inter-organizational integration and collaboration systems hampers business efficiency and optimization for all parties involved in the planning and execution of multi-organization value chain activities: customer requirements for end-to-end tracking and tracing must be satisfied through combinations of human inputs and interventions, heterogeneous information from incompatible ICT systems create barriers to interoperability between network partner systems, and the end-to-end coordination of op-

erational planning and execution activities requires extensive manual effort making network operations costly, non-transparent, error-prone, inefficient and environmentally non-sustainable.

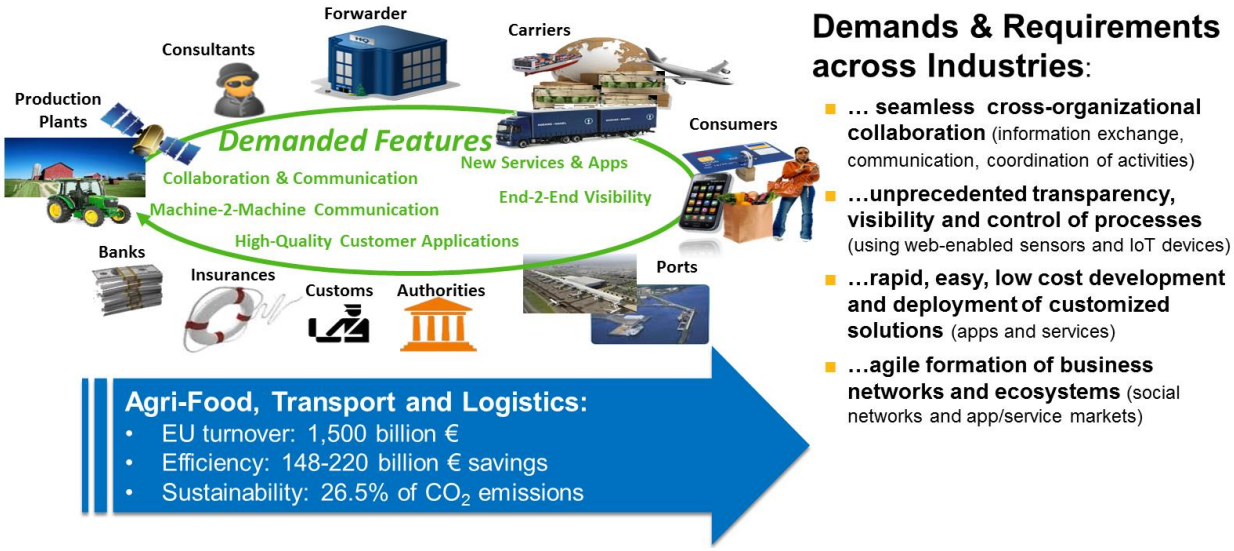


Figure 2: Motivation & Cross-Domain Requirements

2.1.2.2 Business Impact

As depicted above, modern international business – especially in the large and global industries of Agri-Food, Transport and Logistics – is a highly competitive endeavor where resource constraints require organizations to partner with one another to efficiently and effectively address customer needs. In this dynamic market new challenges continually arise, particularly due to increasing customer expectations for personalization and cost reduction. As analyzed in detail in the preceding FI PPP Phase 1 use case projects (SmartAgriFood[2] and FInest[3]), current ICT technologies are either too limited or not capable of properly supporting this evolution of customer requirements. To overcome this and pioneering towards possible future business collaboration platforms, the FIspace will facilitate the following business benefits:

- Better satisfy customer requirements, such as:
  - End-to-end visibility and event management,
  - Enhanced monitoring and tracking of goods as they move along the value chain,
  - Less costly and better tailored offers goods and services,
  - Significantly reduced waste of perishable products,
  - Immediate notification of deviations and the occurrence of hazardous events,
  - Lower environmental impacts through increased network efficiencies, and
  - More transparent operations.
- Substantially increase business efficiency and optimization throughout the entire value chain by:
  - Significantly reducing manual efforts for planning and replanning,
  - Enhancing interoperability among heterogeneous systems based on business standards,
  - Automating support for coordination of operational activity execution,
  - Providing accessibility anywhere and anytime via any device, and
  - Facilitating the rapid identification and contracting of capable business partners.
- Facilitate new business opportunities by:
  - Providing more efficient and transparent service offer management,
  - Optimizing partner contract negotiations,
  - Facilitating new business partner interactions and collaboration opportunities, and
  - Providing access to true end-to-end business and consumer performance metrics.

[2] see project website: http://www.smartagrifood.eu/
[3] see project website: http://www.finest-ppp.eu/

On a broader perspective and with respect to the program-level objectives of the FI PPP, the idea behind the FIspace is to truly move forward in conceiving of a new paradigm in computing that is based on emerging Future Internet technologies and leverages the full potential of the cloud-based services concept [1]. The FIspace pushes boundaries on how business software will work in the future, facilitating innovation and market impact by laying the foundation for adoption by large user groups and external solution providers that can provide additional, novel, and disruptive Apps for the FIspace. In the context of the FI PPP, the FIspace complements the mission of the FI PPP Core Platform Objective: while "FI-WARE aims to provide a framework for development of smart applications in the Future Internet" [2], the FIspace will exploit its technologies for enabling substantial increases in the efficiency and effectiveness of cross-organizational business processes and pioneer novel business models that allow for innovation by external stakeholders with high prospects for industrial uptake and market impact. An important point to note is that the innovative aspect of the FIspace model is the model itself (covering both the technical solution and the proposed business model) – it is not any specific 'technology innovation' such as, e.g., new algorithms, software engineering concepts or the like.

2.1.3 Main Features and Building Blocks

As outlined above, the FIspace will be a Future Internet enabled cloud-based SaaS-platform for enabling the seamless, efficient, and effective business collaboration across organizational boundaries and facilitating the establishment of ecosystems with business benefits for both users from industrial sectors as well as the ICT industry. The FIspace can most suitably be realized on the basis of the Future Internet technologies that are developed in the FI PPP, and will utilize the available Generic Enablers (GE) wherever appropriate. In addition, desirable domain-specific capabilities are being developed to demonstrate the value and capabilities of the FIspace technology and business model (i.e., for inter-organizational process coordination, operational monitoring and tracking, event-driven re-planning, ecosystem application development, monetization, security and privacy management in business networks, etc.).

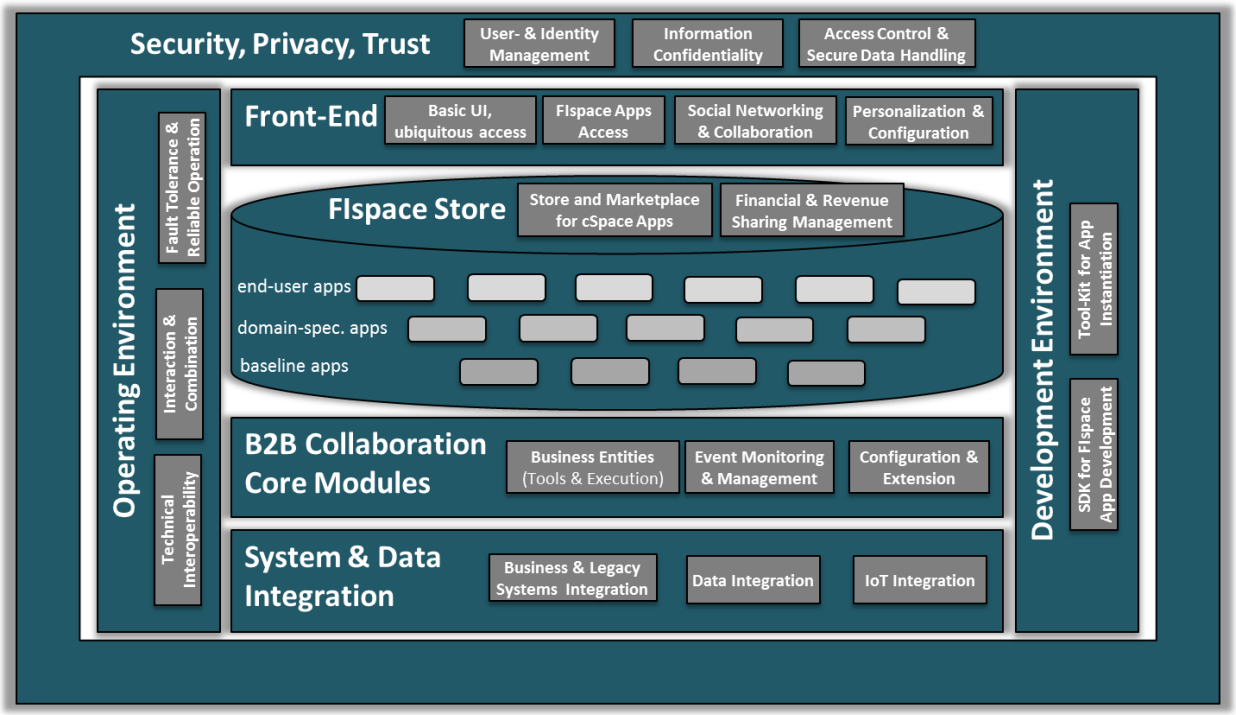


Figure 3: FIspace High-level Conceptual Architecture

In order to realize the envisioned features – i.e., (a) the future support for collaboration in business networks, and (b) a future business model for rapid development of high-quality ICT solutions at minimal costs – the FIspace consists of the following primary building blocks as shown in Figure 3:

(1) The **Front-End** that serves as the ***main point of access for End-Users*** and offers a novel ***'all you need in 1 place' user experience***, including the following main features:
- **Customizable End-User Cockpits**
- **Social Networking and Collaboration Features** for Business Partners and Communities, and support for seamless collaboration on specific business activities and transactions

- **Access anywhere via any device**.

(2) The **FIspace Store** that provides the tool-supported infrastructure for ***providing, finding, and purchasing FIspace Apps*** that provide re-usable IT-solutions for seamless business collaboration and can be *used and combined for the individual needs of End-Users*; for the, the FIspace Store includes:
- The software infrastructure to support the **provisioning, consumptions, purchase, and re-use of FIspace Apps** for both **End-Users** and **App Developers**
- **Financial Management** of the FIspace (pricing, payment, revenue sharing).

(3) The **Business Collaboration Core Modules** ensuring that ***all information and status updates are provided to each involved stakeholder in real-time***, consisting of:
- A **Collaboration Engine** that captures, in form of so-called **Business Entities**, the information that are to be exchanged among collaborating stakeholders along with status and control of the a collaborative business processes, and
- An **Event Manager** that captures and pre-processes both manual (from humans) and automated (from connected systems) events, which trigger the progress and activities of collaborative business processes in a pro-active event-driven manner.

(4) A **System and Data Integration Layer** that allows for the integration and continued usage of existing legacy and business systems as well as the integration of external systems and services, including support for:
- **Connecting business and legacy systems** used by individual users
- **Handling heterogeneous data**
- **Connecting external Systems and Services** (e.g., IoT systems, 3rd party and public services).

(5) A comprehensive **Security, Privacy, and Trust** framework that ensures the secure, reliable, and trustworthy handling of business data making the ***FIspace ‘secure by design’***, including esp.:
- **Access Control and Identity Management** for all users
- **A set of Security Mechanisms** to ensure information security, attack prevention, etc.
- **Developer support** to ensure correct usage of necessary security mechanisms in FIspace.

(6) An **Operating Environment** that ensures the ***technical interoperability of FIspace Components and Apps*** and the ***consistent behaviour of the FIspace***, including:
- **Technical Interfaces and Protocols** for the FIspace Components and Apps
- **An Enterprise Service Bus** to support the interaction of FIspace Components and Apps
- **Replication and Consistency Services** to ensure fault-tolerance and transaction support.

(7) A **Development Toolkit** providing tool-support for the development and instantiation of FIspace, particularly for:
- **App Developers** to support the development and provisioning of Apps in accordance to the technical governance and procedures of the FIspace
- **Business IT Experts** who customize and extend the FIspace to the individual needs of End-Users at an individual or organizational level.

While referring to Section 4 where the initial technical design and development plans for each of the FIspace components is presented in detail, the main features of the FIspace in summary are:

- An open application that can be extended and customized for specific stakeholder requirements through the use of Apps that can be “mashed up” (integrated) using services of the FIspace (an extension of the app concepts of mobile telephone providers such as Apple, Google and Microsoft);
- An integrated Front-End as a central point of access to all features and Apps, enabling an “all you need in 1 place” user experience with customizable views for individual actors;
- Integrated novel technologies that provide the basis for future collaboration in business networks, enabling “all information to each stakeholder in real-time” as well as the seamless interaction and coordination among business partners;
- Information integration from legacy and third party systems enabled through a service-based integration layer that is enabled and supported by FI-WARE Generic Enablers;
- Modern technologies that ensure the secure, reliable, and trustworthy handling of business information in the Cloud, and the consistent operation of the FIspace;
- An (App) Store that facilitates the provisioning, consumption, and marketing of novel on-demand solutions for B2B collaboration, along with an integrated toolkit for developers.

2.2 FIspace Operation Model – Illustrative Example

While the overview above remains on a general level, the following explains the intended operational model of the FIspace at hand of an illustrative example. For this, we here use the ‘Spraying Advice’ scenario as the sample use case: taken from the Greenhouse Management & Control trial (cf. T422), this has served as the initial example discussed at the project kick-off meetings and has then been taking up within WP200 for the scenario-based technical design of the FIspace components and their consolidation. It is important to note that the scenario and FIspace Apps as described below are a simplification only intended for illustration and technical design work; the actual scenario analysis and planned FIspace-based solutions for this are elaborated in WP400 and described in detail in Deliverable D400.1.

2.2.1 Sample Scenario: Spraying Advice

The sample scenario considers a situation where a farmer (here: Franz Farmer) demands a spraying advice from an expert (here: Ed Expert) in order to handle a disease on tomatoes in his greenhouse; as an additional stakeholder, a local state authority is involved who needs to approve the usage of the pesticide recommended by Ed because of its environmental impacts. As indicated above: this scenario and the actors are fictitious and only intended for illustration purposes.

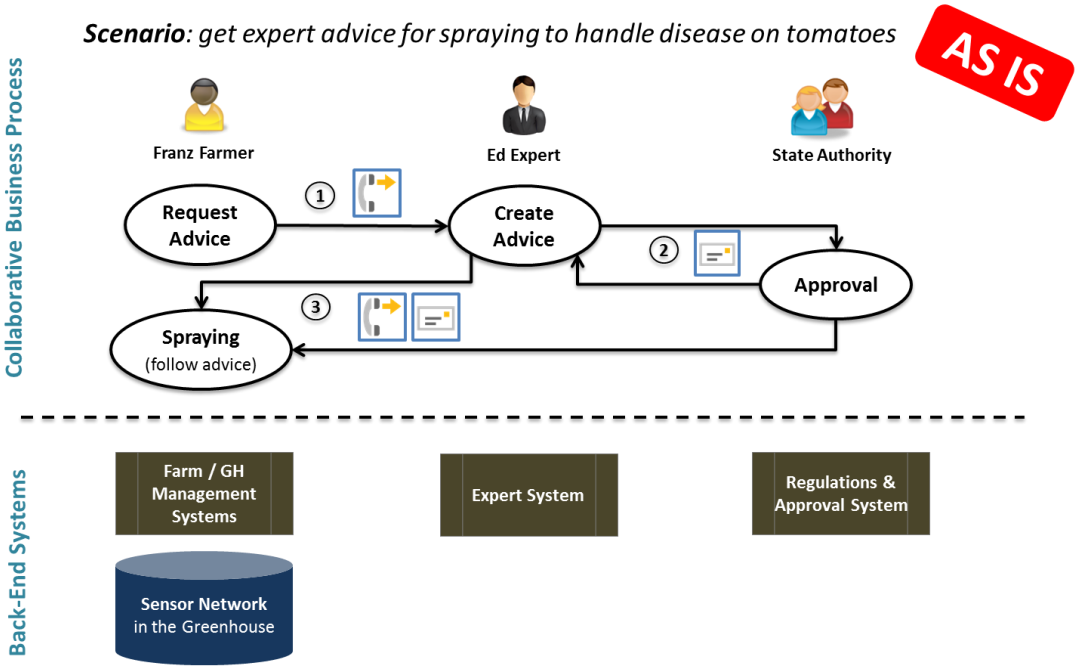


Figure 4: Sample Scenario 'Spraying Advice' - AS IS Overview

Figure 4 above provides an overview, indicating the current situation (AS-IS): Franz calls up Ed and explains the situation. To be able to provide adequate advice, Ed needs more detailed information on the tomatoes and the current situation; for this, commonly Ed needs to visit Franz’s farm onsite in order to to investigate the disease and get the relevant data from Franz’s farm management system (product details) and the sensor network in the greenhouse. After Ed Expert has concluded the advice (using, among other sources, his own expert system), he files the request for the necessary state approval by letter; the authority approval is finally given again by postal letter to Ed and Franz.

This creates a lot of overhead, creates costs, and delays the time until Franz can undertake the necessary actions to handle the disease. The purpose of FIspace is to overcome this by enabling to conduct all necessary interactions ‘online’ where the involved players conduct the business interaction via Apps that are connected to the relevant on-premise systems and inform each actor about the current status of the collaboration business process in real time. Figure 5 below outlines how the TO-BE scenario shall look like: all involved stakeholders perform the information exchange and communication between them via the FIspace, using dedicated FIspace Apps that, for instance, allow Franz to send his request for advice directly to Ed via the FIspace and provides Ed with remote access to the greenhouse sensor data, and also the spraying approval by the state authority is requested and processed ‘online’ via the FIspace. In addition, the stakeholders’ respective back-end systems are connected to the FIspace, so that the information will needs to be exchanged among the stakeholders can be imported into the FIspace Apps,

which allows decreasing manual effort and quickly setting up individualized solutions for collaboration business activities with full control on the information sharing by each participating business partner. With this, the FIspace enables the seamless, efficient, and easy business collaboration and eventually will facilitate the business benefits as outlined above.

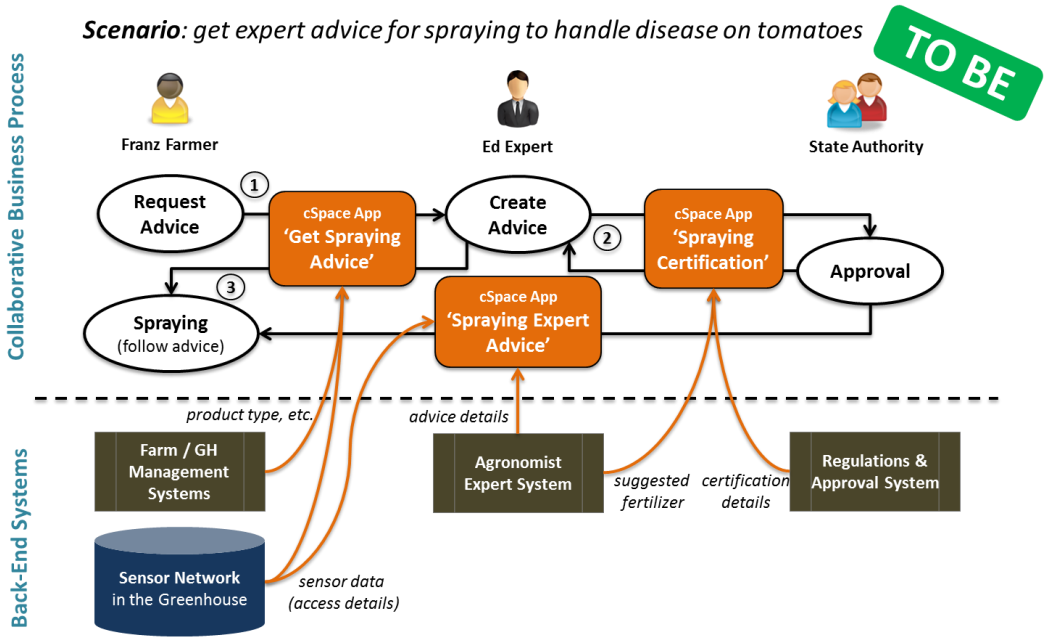


Figure 5: Sample Scenario 'Spraying Advice' - TO BE Overview

2.2.2 Illustration of FIspace Apps and Overall Operational Model

To illustrate how the TO BE Scenario shall actually be realized with the FIspace, let us first investigate a typical FIspace App. For this, Figure 6 shows an illustrative design of the 'Get Spraying Advice' App that is used by Franz Farmer. It consists of three main functionalities: on the left hand side, Franz can prepare the Request for Advice (incl. basic product information, a description of the request, and a link to the IoT-enabled sensor network in his greenhouse), and select the receiver of the request from the business contacts that Franz maintains in the FIspace. Once the advice is prepared, Franz can receive it and optionally conduct additional interactions with the business partners (see right hand side for some initial ideas). An important and novel feature of the FIspace is that Franz can always see the actual status of his request (see bottom): this is automatically updated whenever a partner undertakes an action (e.g. when Ed Expert has finished the preparation of the advice, or when the state authority has provided the approval).

Figure 6: FIspace App 'Get Spraying Advice' – Illustrative Design

With this kind of FIspace Apps, each involved actor can see all relevant information in one place and is informed on status updates in real-time, especially on critical situations such as delays or other deviations that often are business-critical and hence are demanded to be known as quick as possible. In addition, such FIspace Apps can be developed rather quickly, which allows businesses and partner networks to promptly adapt their collaborative IT-infrastructure with respect to newly arising market opportunities.

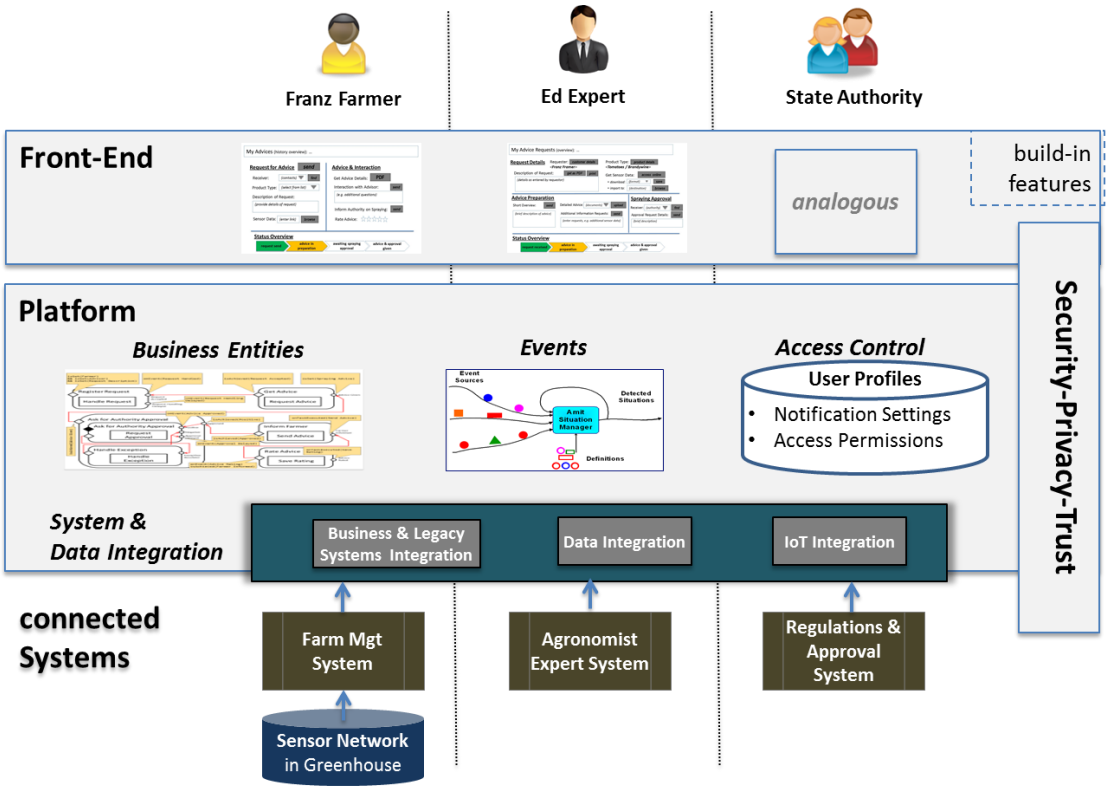


Figure 7: Illustration of FIspace Overall Operational Model

Figure 7 provides a visualization for explaining how the FIspace is intended to work. When the stakeholders log on to the FIspace, they can see their new notifications (similar to 'new messages' in e.g. facebook) and access their respective FIspace Apps: e.g. Franz the one outlined above, Ed Expert an analogous one that supports the creation of advices and seamless interaction with his business partners, and another one for the State Authority to handle spraying approvals. The execution of the cross-organizational interaction is facilitated by the generic FIspace platform-features, in particular by:

(A) The *Business Entities* that represent the collaborative business process and control its execution by managing the interaction and control flow between the associated apps, for example:
- When Franz sends out a 'Request for Advice' via the FIspace App from Figure 6, the underlying Business Entity triggers a process step which ensures that Ed Expert is immediately notified about a new request within his personal FIspace Front-End, resp. FIspace App;
- When the State Authority provides the spraying approval, both Franz and Ed are notified immediately within their respective Front-End / Apps.

(B) The *Event Engine* that captures and processes various types of events that are used to e.g. trigger the progression of Business Entities or alert End-Users on critical situations, therewith making the FIspace an event-driven platform; examples for events are e.g.:
- A new 'Request For Advice' send out by Franz or another Farmer is an event, which triggers the progress of the Business Entity as described above, or
- Franz can get notified on external events that maybe critical for his business (e.g. temperature in the Greenhouse reaches a threshold, or an upcoming thunderstorm),

(C) The *Access Control* which ensures that only those people get access to information who have the respective permissions and nobody else (ensured by the all-embracing Security-Privacy-Trust framework of the FIspace), and also the reception of notifications in accordance to the personal configurations and settings of the user's profile in FIspace.

(D) The ‘System & Data Integration’ facilities that allow to quickly set-up connectors or adapters to existing system landscapes using standard technologies and data formats so that Franz, Ed, and the State Authority to connect their respective back-end systems to the FIspace, e.g.:
- Franz can grant Ed access to the IoT-enabled sensor network in his greenhouse via the FIspace (similar to e.g. sharing a link or video with friends in facebook), or
- The State Authority can upload details on the spraying approval from its existing in-house regulations and approval management system landscape.

Of course, also the other technical building blocks as identified above in Section 2.1.3 are important and needed in order to ensure a proper execution and usage of the FIspace: The *Operating Environment* ensures the technically smooth interoperation of all components and also the fault-tolerant execution, which is of high importance when dealing with business-relevant data and processes; the Development Environment provides the tool support for the developers of FIspace Apps and ensures that the implementations comply with the FIspace standards for security and technology, and as the marketplace for FIspace Apps the *FIspace Store* provides the basis for the economic exploitation of the FIspace.

2.3 The FIspace Usage Model

As the final part of the FIspace overall design, the following outlines principal usage model of the FIspace. For this, we first define the main processes for using and working with the FIspace, and then define the workflows for the main user groups.

2.3.1 Main FIspace Processes

The FIspace processes define how the features provided by the FIspace shall be used to conduct business activities. In the context of a comprehensive conceptual overview, we primarily focus on the processes from the user perspective for using and working with the FIspace; of course, more detailed processes and interactions between the components are required for the technical realization.

Figure 8: Main FIspace Usage Processes

The above figure summarizes the main FIspace usage processes, and indicates the technical building blocks that will realize the respective functionalities. The first process is concerned with two aspects: firstly, all end-users (i.e. experts working for a company) are demanded to be registered on the FIspace, and their accounts with access permissions and app usage are managed on the company level; further details on this are provided in Section 4.1. Secondly, the management of business partners and contacts shall

be supported by the FIspace as a single point where all relevant information are available, which is commonly considered as a highly value-adding feature of a business collaboration platform; for this concepts from collaboration platforms from the B2C world (e.g. facebook, google+) as well as from the B2B world (e.g. LinkedIn, Yellow Pages concepts, etc.) will be analyzed to provide best-value business network management services without violating confidentiality interests of the participating companies. The processes (2) – (4) are concerned with the development, purchase and usage of FIspace Apps, which shall provide the value-added services for collaborating business networks with integrated collaboration features as outlined in the illustrative example above: at first, a FIspace App is implemented, using the Development Environment; then, it is uploaded to the FIspace Store, where interested user can purchase it, and finally it is set-up for the individual user, including the connection to existing system landscapes and the configuration for the individual needs of the end-user, e.g. personalization of appearance, notification settings configuration, and the creation of customized solution by mashing up several FIspace Apps.

While these processes outline the steps for creating and using a single FIspace App, the overall aim is to enable and foster the efficient and seamless collaboration in existing, emerging, and growing business networks. For this, several and an extensible number of FIspace Apps shall be developed for the same (or at least similar) collaborative business process, so that the participating businesses can constantly extend their partner network and adapt their collaborative IT solutions.

2.3.2 Principal User Groups and Workflows

In order to illustrate outline how the FIspace shall work in more detail, the workflow descriptions in the tables below outline the usage procedures for the following principle user groups of the FIspace:

1. **End-Users**: the business experts using the FIspace to conduct the daily business activities, with special focus on their interaction and collaboration with business partners;
2. **Business Process Engineers:** the ICT experts (internal or external) that support End-Users in the configuration of the FIspace for their individual business needs, particularly for the definition of customized business processes by using the FIspace Apps and the FIspace's customization support services (on various levels: company / organizational unit / individual);
3. **App Developers**: the software and system providers who offer solutions and applications in form of Apps via the FIspace.

Of course, the actual ecosystem of the FIspace will encompasses much more roles whose respective business interests and motivation need to addressed and supported in order to ensure acceptance in a broader community. Examples for additional roles are e.g. the more detailed separation of the End-User group into e.g. Producers, Shippers, Logistics Service Providers, etc., who will together form the customers and therewith drive the usage of the FIspace; also, several additional roles on the IT-provider side can be distinguished, such as e.g. the Platform Operator(s), Infrastructure Providers, Consultants, and 3rd-Party Solution providers. This is subject to the business model for FIspace, which will be elaborated in close interaction with WPP500 (esp. T520).

Table 1: Workflow End-User

Step	Activity	Used FIspace Components

1	**Registration & User Profile Maintenance**
	• Register & Log-In to FIspace
	• Create & Maintain User Profile (individual / organizational unit / company level):
	  ○ Select / define role
	  ○ Provide basic profile information
	• Personalize Cockpit (appearance, basic notification & communication settings)
		Front-End
		• Registration & Log-in Process
		• User Profile Management
		• Personalization Features
		Security (indirect, integrated in Front-End)
		• Secure Log-In & Usage
		• Access management

2	**Find & Manage Business Partners**
	• Find business partners (known & unknown)
	  ○ Via public user profiles
	  ○ Via offered business services
		Front-End
		• Search for public user profiles
		• Basic Contact Management
		• Basic networking & collab. features

	• Manage your business contacts
	  ○ Maintain contact data
	  ○ Seamless communication via social networking & collab. features
	• Manage business partners
	  ○ Set-up & manage contracts
	  ○ Rate partners (public + private)
	• Create Business Communities (for e.g. areas of interest, establishing networks, …)
		• Formation of communities
		Apps (purchased via Store, access via Front-End), e.g. for
		• Advanced business partner mgt
		• Adv. business partner search
		• Adv. networking & collab. features
		• Partner Rating & management
		• …
		Security (indirect)
		• Information Security
		• Access management

3	**Get & customize Apps for Business Activities**
	• Find & get need Apps
	  ○ Available in FIspace Store
	  ○ Pre-configured by Business Process Engineers
	• Customize for individual needs (only ‘personal configuration’, basic configuration / extensions / combination by IT experts / Business Process Engineers)
		Front-End
		• Personalization & Configuration for Apps (selection, personal appearance, look & feel)
		• Access Apps (consume Apps via UIs that are provided in Front-End)
		App Store
		• Find & investigate available Apps
		• Purchase Apps
		• Use pre-configured Apps / cust. solution provided by associated IT Experts / BP Engineers

4	**Execute Daily Business Activities** Use Apps & FIspace Features for conducting daily business, incl.: • Define specific notification & comm. settings for business activities • Use collaboration features for specific business activities and transactions • Continuously manage business partners	**Front-End** • Access to Apps (UIs integrated in Front-End) • Personalization & Config. Features • Embedded social networking & collaboration features (basic + advanced) • Access to statistics on App Usage, Business Partner Information, etc.

Table 2: Workflow Business Process Engineer

Step	Activity	Used FIspace Components

1	**Find & get relevant Apps**
	• Search / browse FIspace Store
	• Investigate for suitability
	  ○ Features & functionality
	  ○ Interfaces, Data Structures
	  ○ Options for configuration, extension, re-sue
	  ○ Pricing & payment models
	• Purchase relevant Apps (for company / org. unit / individuals)
		App Store
		• App Search & Discovery facilities
		• Investigation Support for Consumers (features + pricing models) and for Developers (technical details)
		• App Purchase Support
		• Ratings of Apps by FIspace Community

2	**Create customized Solution**
	• Design desired Workflow for End-Users
	  ○ Sequence / Process of Apps
	  ○ Data models and relevant systems
	  ○ Interaction & collaboration with business partners
	• Configure / extend / define data structures and technical workflow over Apps
		Development Toolkit (encompasses various developer tools)
		• Configuration / Extension for Collaboration Artifacts & Event Handling Rules
		• Customization of Apps (configuration, extension)
		• Mash-Up & Orchestration of Apps

	○ Configure / extend Collaboration Artifacts + Event Rules (or define new / additional ones) ○ Configure / extend selected apps (hide / rename data fields, resp. add additional functionality) ○ Orchestrate Apps into desired workflow: define / ‘mash-up’ execution sequence & technical interaction models • Connect relevant systems (‘legacy’ systems and external systems & services) and integrate them into the customized workflow ○ Define & create connectors to between FIspace Apps and systems ○ Define ‘data mediators’ to integrate data systems <-> FIspace	**System & Data Integration** • Tool-supported techniques for connecting business systems (legacy & standard systems, ...) • Tool-supported techniques for connecting external systems & services (e.g. IoT-enabled sensor system, 3rd-party services) • Data mediation & integration facilities
3	**Provide customized solution to End-Users** • Provision to relevant End-users by making the customized solution accessible in the personal Front-End of relevant End-Users • Pre-configure for End-Users: Access Rights, setting for notifications + communication • Configure pricing & payment models (for company / org. unit / individual level) • [optional] publish re-usable parts of customized solution in FIspace Store	**Front-End** • Personalization & Configuration for individual End-Users • Configuration (Access Rights, Notification & Comm. Settings) **Store** • Configuration of payment models • Publication of Apps

Table 3: Workflow App Developer

Step	Activity	Used FIspace Components
1	**Find existing Apps to build upon** • Search / bowser FIspace Store • Investigate for suitability & re-usability ○ Features & functionality ○ Interfaces, Data Structures ○ Options for configuration, extension, re-sue ○ Pricing models, terms & conditions for re-use • Buy Apps for re-use (‘development license’)	**Store** • App Search & Discovery facilities • Support for Detailed Investigation (features, functionality, technical details, pricing models, terms & condition for re-use • App Purchase Support for re-use • Ratings of Apps by FIspace Community
2	**Develop App** • Using ‘FIspace Development Toolkit’ to develop app to comply with the FIspace ‘technical governance’ incl. in particular: ○ UI Framework & Technology ○ Technical Interfaces & Interaction Protocols ○ Usage of security, privacy, and trust technologies • Configure / extend / define data structures and define interaction with re-used Apps ○ Configure / extend Collaboration Artifacts + Event Rules (or new / additional ones) ○ Configure / extend selected apps (hide / rename data fields, resp. add new functionality) ○ Define technical interaction with re-used Apps (‘orchestrate’ or ‘mash-up’) • Prepare for re-use and for integration with legacy & external systems	**Development Toolkit** (incl. developer guidelines, suggested development tools) • Compliance with FIspace ‘technical governance’ (UI technology, technical interfaces & interaction protocols, security techniques) • Configuration / Extension for Collaboration Artifacts & Event Handling Rules • Customization of Apps (configuration, extension) • Mash-Up & Orchestration of Apps

	○ Define interfaces for connecting systems ○ Define supported / expected data structures	
3	**Publish App on FIspace** Publish new App in FIspace Store, incl. • Creation of App Description • Configure / define pricing models, usage terms & conditions • Conduct FIspace App Publication Process	**App Store** • Publication Process for Apps • Configuration / definition of pricing models, usage terms & conditions • ‘Technical Governance’ Check

3 Release Plan and Development Methodology

The goals of FIspace are to drive development and trials for an integrated and extensible business-to-business collaboration service and associated Apps, thereby establishing *the* standard for supporting and optimizing inter-organizational business collaboration in the global transport, logistics, and agri-food industry. To achieve its goals within the 24 month timeframe, FIspace follows an *incremental, iterative development and release approach.*

This means that FIspace results will developed along 8 consecutive 3-month milestones. Each of those quarter-year milestones defines delivery and synchronization points for major outcomes. Within each quarter-year time frame, agile methods are applied for managing the delivery of those outcomes.

This means that FIspace follows a more “traditional”, structured process for deliverables along those eight milestones, where deliverables are related to WPs and Tasks. Complementing this more “traditional” process, FIspace develops frequent releases of results in dedicated teams for each FIspace component. This agile approach foresees that each 2 weeks an increment of software is delivered.

This approach is justified as follows:

- Providing early and continuous releases of specifications, technical interfaces and implementations of components ensures that domain Apps will be developed and made available in the FIspace app store early on in the project, thus facilitating the involvement of interested domain players outside the consortium.
- Early availability of working software will stimulate interest from open call members, such as to obtain development partners (including SMEs) to begin rapid development of domain Apps so that the foundation for a robust ecosystem for the targeted FIspace domains (transport, logistics and agri-food) can be established, as well as to build a critical mass for large scale expansion in Phase 3.
- The incremental release of WP200 results will allow for a continuous, incremental validation approach of the FIspace solutions, starting with software testing of the FIspace components, validation of the FIspace services in an experimental environment, up to the in-the-field deployment of FIspace software as part of trial and demonstration sites.
- Combination of more “traditional” project management with agile principles will ensure structured and effective technical management (including continuous identification and mitigation of risks), while remaining agile to respond to novel findings (e.g., from the trial experiments), emerging requirements (e.g., from the inclusion of open call members), and technology developments (e.g., from new releases of Generic Enablers).
- The incremental approach leverages experiences gained from integrating and aligning with the Technology Foundation (“Core Platform”) activities (including FI-WARE and its successor project), specifically for what concerns the use, integration and validation of its generic enablers.

Below, we will first describe the overall software release plan that is aligned with the eight project milestones, (Section 3.1) and then will elaborate on the agile approach followed in FIspace (Section 3.2).

3.1 Overall Release Plan

Overall, the project work plan consists of **eight milestones** that define the verifiable results that denote the expected progress and thereby facilitate decision making concerning the next stages of the project.

Each development and validation cycle involves technical development of FIspace components (WP200) and Apps (WP450), as well as their deployment (WP300), followed by use in trial experiments (WP400) and complemented by dissemination and open collaboration (WP500).

Overall **three** such **major development and validation cycles** are planned for FIspace.

- **Development and Validation Cycle 1 (ending in Mo 12 – MS4 with the Aztec Release of FIspace)**: Provides the first stable results of the initial software release and trials of FIspace together with the release of a set of baseline Apps.
- **Development and Validation Cycle 2 (ending in Mo 18 – MS 6 with the Inka Release of FIspace)**: Provides results of the second release and trials of FIspace, together with domain Apps, contributed jointly with open call members.
- **Development and Validation Cycle 3 (ending in Mo 24 – MS 8 with the Maya Release of FIspace)**: Provides the final release of FIspace and the final results of the cross-domain trials, including extended and refined Apps.

In addition to fostering the synchronization and timely delivery of outcomes, the timing of milestones has been aligned with the **overall FI PPP programme schedule**, i.e.,

- **MS2 (due Mo 6)** will already provide a comprehensive, **technical specification of FIspace**. This will allow FIspace to start working and interacting with the external developer community early on, thereby fostering building the necessary mass of App developers and interested stakeholders to incubate the FIspace ecosystem. More specifically, this early release constitutes **instrumental information for Phase 3 proposers**, allowing them to align their efforts with the pre-investments made in Phase 2 of the programme.
- **MS4 (due Mo 12)** provides the first stable release of FIspace which already underwent a first trial and experimentation round. It will thus constitute a reliable platform for Phase 3 projects to interact and build upon.
- **After Mo 12**, the Phase 3 efforts are supported by FIspace through dedicated knowledge transfer and training activities (including workshops).
- **MS8 (due Mo 24)** will provide sustainability plans and strategies to ensure continued support and maintenance of FIspace even after the project has ended, and while Phase 3 projects will run for one further year.

The below table depicts the main software releases together with the relevant milestones of the project, i.e., linked to the milestones of the App development, which builds on top of the platform. Section 4 below details the individual FIspace components and describes which features can be expected during the three major platform releases.

Table 4: FIspace Platform releases and Alignment with App Development

Release	Milestone	FIspace Platform	FIspace Apps
Specification	MS1: **Consolidation** (M 3)	Consolidated conceptual design; Detailed release plan; Development support facilities set-up	Release and development plan for Apps
	MS2: Specification (M 6)	**Public release of FIspace specification (technical design)**	
Aztec	**MS3: Release V1** (M 9)	**1st release of FIspace core feature prototypes ready for trials (internal)**	**1st release of Apps** (baseline Apps) Identification of requirements for domain Apps
	MS4: Trial-Round 1 & Large scale expansion (M 12)	Maintenance updates of FIspace V1	
Inka	**MS5: Release V2** (M 15)	**2nd release of FIspace (public)**	**2nd release of Apps** (incl. Apps from open call members)
	MS6: **Trial-Round 2** (M 18)	Maintenance updates of FIspace V2	
Maya	MS7: **Release V3** (M 21)	**3rd and final release of FIspace (public)**	**3nd release of Apps** (incl. apps from open call members)
	MS8: **Trial-Round 3** (M 24)	Maintenance updates of FIspace V3	

3.2 Agile Methodology

As motivated in the introduction to this section, FIspace needs to be able to efficiently and quickly realize the planned development, experimentation, and outreach activities, to ensure their proper alignment by continuously validating implementations with respect to user and stakeholders, and quickly act on detected faults and change requirements. In order to achieve this – and as already defined in the DoW – the project consortium has decided to establish an agile working methodology throughout the whole project. For this, we adapted the methods from Agile Software development that are well-established in industry and allow for quick, iterative, and well-aligned execution of software development that is required for the FIspace project, in particular concepts from Lean Software Development [3], Design Thinking [4], and SCRUM [5] are adopted.

As in every project, these methodologies cannot (or at least should not) be taken 'off the shelf', but need to be adapted to the specific set-up and characteristics of the project; in addition, the methodological set-up is subject to change and evolution throughout the duration of the project, meaning that the bodies, communication channels, and interaction patterns are continuously refined and adjusted with respect to the arising needs and gained experiences. The figure below illustrates the initial agile methodological set-up that has been established in the first months of the project, and complements the formal project management structure as described in detail in Deliverable D100.1. The overall aim is to facilitate the necessary interaction and communication across the work packages and the respective task teams in an easy, simple, and quickly adaptable manner. For this, the main interaction channels are as follows: the trial teams in WP400 define the requirements and provide feedback to the development teams (mostly in WP200 where the generic FIspace software infrastructure is developed, but also in T450 where the FIspace Apps are developed). The development teams take the demands and feedback into account for the design and implementation of the technical solution, which is then provided via deployment on the Cloud infrastructure hosted by WP300; the WP300 team also provides the technical infrastructure whereupon the trials can perform the experiments (cf. T340). Finally, the Business & Exploitation Team in WP500 takes up the results from both the trial and the development efforts for dissemination and stakeholder engagement, and provides necessary inputs and feedback on business models, industrial adaptation, and IPR management.

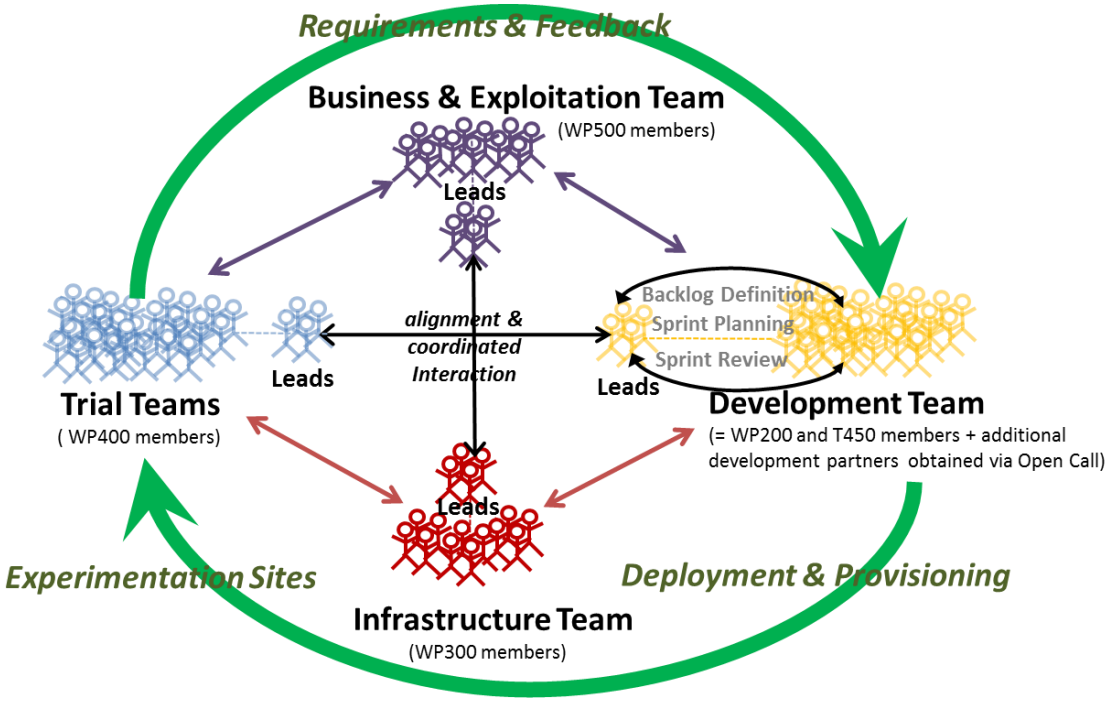


Figure 9: Agile Project Methodology

Obviously, this highly interdependent project activities demand an efficient, easy, and continuous communication among the teams, best directly between the experts in the respective work packages and tasks. In order to ensure this, the initial methodological set-up is based on the following basic principles:

(1) Bi-weekly delivery cycles to ensure timely progress and, more importantly, quick & immediate feedback,
(2) Direct and loosely coordinated communication between teams and experts to not lose time via boards and longer decision making procedures, and
(3) Regular communication among all project members by e.g. internal education sessions or all-hands meetings where latest results are presented.

The interaction is facilitated by the respective task leads in close alignment with the WP leads and the formal project management bodies, and then executed among the experts directly. In addition to this, several cross-WP activities are driven by assigned groups or task forces, such as e.g. the alignment with the FI PPP, the validation strategy for Generic Enablers and interaction with FIWARE, or the modeling of Business Entities and Event Handling Rules for the trials which represent the collaborative business processes and form a foundation of the pilot apps. The initial feedback from project members on this methodology is very positive, but also various aspects for improvement have been determined, which shall be addressed within the upcoming work cycles.

Specifically within WP200, it has been decided to establish a SCRUM-based methodology for the technical design and, later on, for the development work. The work package runs in 2-weekly sprints where the WP lead takes over the role of the Chief Product Owner (= defining WP-level goals for next sprint and / or milestone), the Task Leads takes the role of the Product Owner (= defining goals of next sprint, ensuring and reviewing the progress, and being responsible for the ‘product’), and the Task Members define the actual work plan for achieving the sprint goals, i.e. defining detailed actions items and necessary interaction with other teams. While in the beginning the sprint planning has been conducted by spreadsheets, the set-up of a professional tool-supported software development environment is in progress which, among a distributed code repository and deployment management infrastructure also contains sophisticated SCRUM tool support.

4 FIspace Components Initial Technical Design

This section presents the initial technical design of the 7 main technical components that are developed in WP200, therewith representing the main part of this deliverable.

As shown in Figure 10, the FIspace components are the Front-End (developed in Task 220), the FIspace Store (T230), the B2B Collaboration Core Modules (T240), the System & Data Integration facilities (T250), the Operating Environment (T270), the Security-Privacy-Trust framework (T270), and the Development Environment (T280). The following presents them in this order, and for each one we define the planned features along with a release plan, the Generic Enablers that are planned to be used as well as additional technology choices for implementation, and an initial technical architecture.

It is important to note that this presents an initial conceptual technical design that – in accordance to the agile methodology as outlined above in Section 3.2 – shall be refined, revised, and (where necessary) changed in the course of the project with respect feedback from the trial teams and app development activities in WP400 as well as from external stakeholders and experience. Moreover, it is to note that the following presents the initial version of a thoroughly consolidated technical design of the FIspace, with the primary goal of ensuring that all necessary features and functionalities are included and a clear relationship among the components is defined. This has been achieved by

- Continuous interaction with all project partners, started at the kick-off meetings in the beginning of April 2013 and continuously continued
- The SCRUM-based working model for WP200 with bi-weekly sprints
- Several technical meetings, including a 2-day technical team workshop in May and various virtual and physical meetings of task teams and on other special topics.

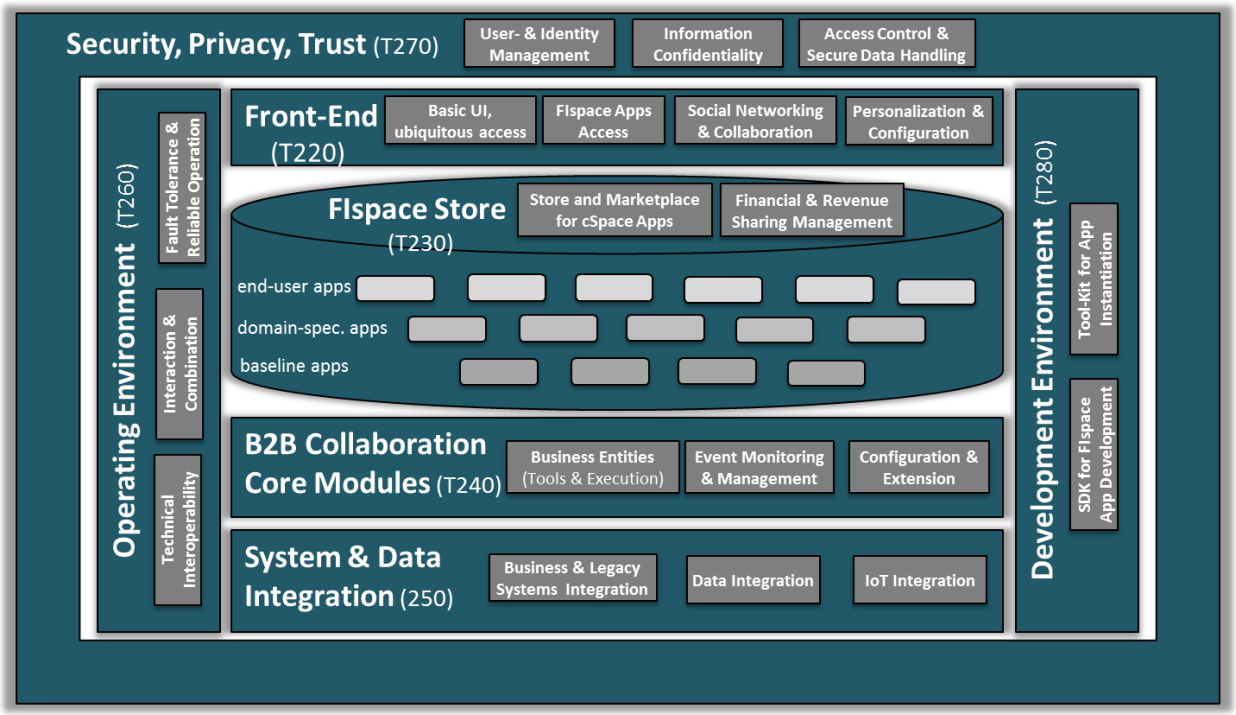


Figure 10: FIspace High-Level Technical Architecture with WP200 tasks

4.1 The FIspace Front-End

The Front-End builds the main access point for end-users of the FIspace platform, whereas this is not limited to the user interface, with which the end-users directly interact, but also encompasses server-side components that are closely related to the user interface. Through the integration of external user interfaces (e.g., from the store, externally developed Apps or other external providers), the Front-End facilitates an ‘all you need in one place’ user experience and create a central access point. To support the diversity of FIspace users and devices the Front-End user interface will adapt to specific needs. This allows ubiquitous access and is enabled by backend components of the Front-End. Beyond the adaptation

to different devices, the Front-End also supports the configuration of the user interface. This allows the interface personalization in order to address specific user needs or enable custom brandings for companies. The Front-End also enables users to create relations to business partners to facilitate the communication among them (comparable to modern social networks). The Front-End will also visualize such relationships and by this, facilitate the analysis of business networks.

In this section we describe the FIspace Front-End component. First we give a general description of it with all its main features and building blocks. These building blocks are based on the Sub-Tasks of the Task T220, as stated in the DoW. Each Sub-Task is addressed in a separate subsection, whereas every subsection contains an overview and a description of the main features, a first technology survey containing the review of potential Generic Enablers and a preliminary release plan for the most significant features. In the end of this section we provide the preliminary high-level technical architecture of the Front-End component.

4.1.1 End-User Core Front-End Development

In this section we describe the main features and first results of Sub-Task T222 – End-User Core Front-End Development. This encompasses a general overview, a description of the main features we elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for an implementation and a preliminary release plan for the coming three releases of the Front-End component. We address each of those aspects in a separated subsection below.

4.1.1.1 Overview & Main Features

One of the main features of the core is the definition of the basic layout and look & feel of the user interface. In Figure 11 we show a first mock-up for this. The top area contains the content that is static for the user interface, whereas the bottom area holds the variable content, which depends on the selected menu item. The static elements encompass the name of the logged in user, a profile picture, a search bar, the FIspace logo, event icons and the menu bar. The mock-up indicates recent news and important messages as the content for the variable part if the *Home* menu item is selected.

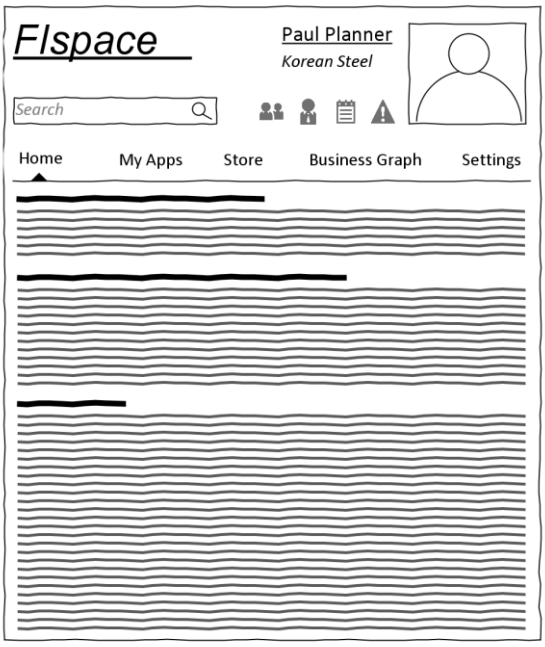


Figure 11: Basic Layout of FIspace User Interface (Person Profile)

Another main feature for the Front-End is the provisioning of user interfaces for the login of existing users and the registration of new users. Figure 11 gives a first impression of how this can be done. The layout follows the same approach as presented in the previous figure: the upper part is static and displays fields to login or register, whereas the part below is dynamic and provides different general information about the FIspace. As indicated in the right figure, the registration only requires very basic information (quick register). The user can complete his or her profile after the login.

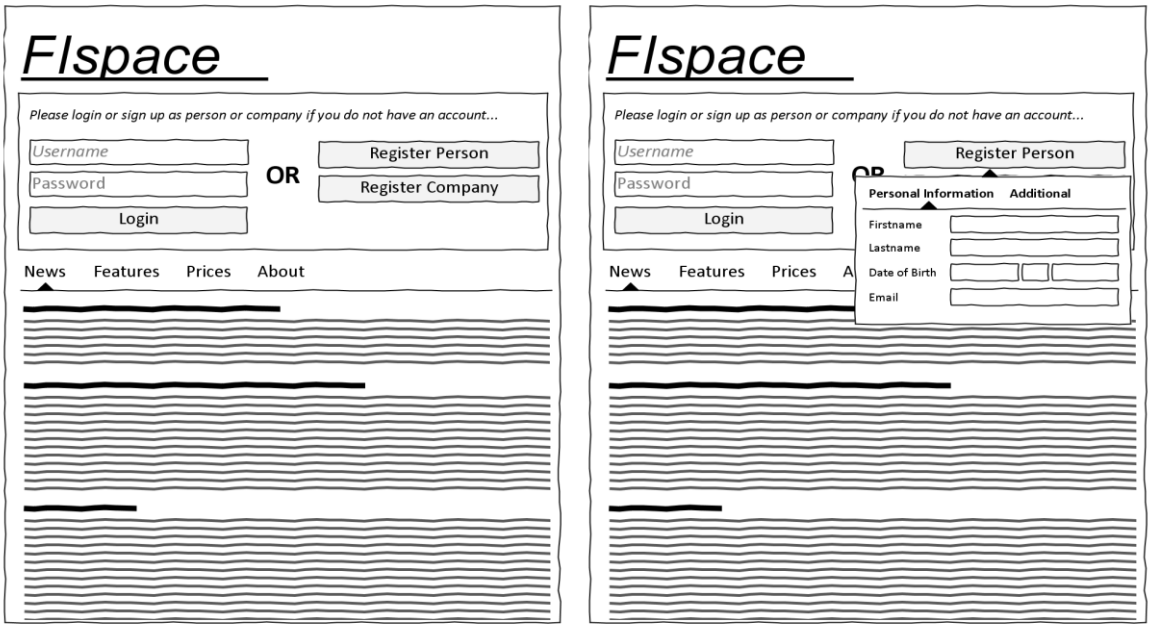


Figure 12: Login and Registration for User and Company Profiles

Important to note is that the Front-End distinguishes two different types of profiles. The first type is already shown in Figure 12 and is called *Person Profile*[4] and is foreseen for individual persons. The second kind is called *Company Profile*[5] and shall be used by organizations or companies to represent their entities in the FIspace. This kind of profile can be administrated by several users and provides additional capabilities (e.g., employee management or app permission management). Figure 13 shows a mock-up for the company profile and the employee management.

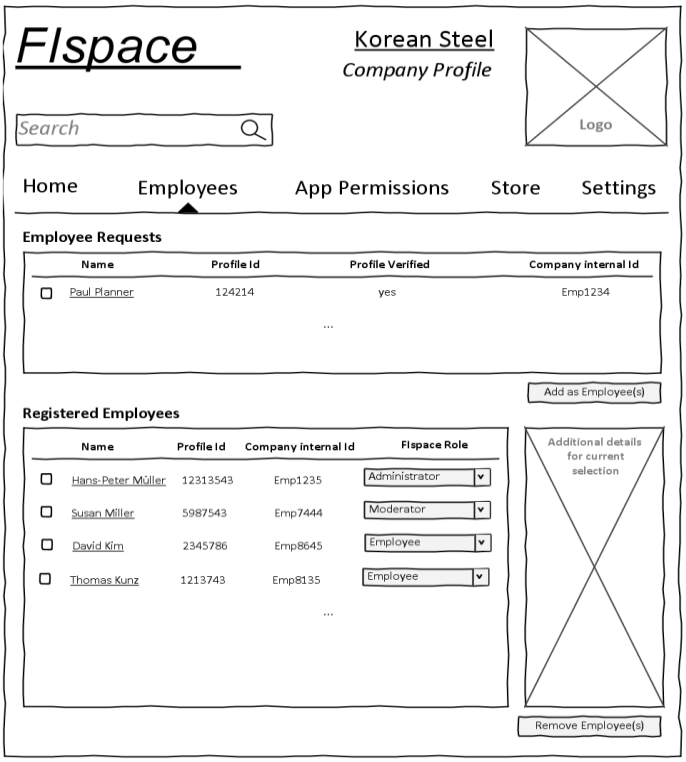


Figure 13: Company Profile with Employee Management

[4] Note: this is a working title
[5] Note: this is a working title

The FIspace platform allows Apps to send notifications about changes in the executed business processes to the Front-End and its user interface(s). How such notifications can be presented to the user is shown on the left side of Figure 14. In order to enable users to define how specific notifications shall be displayed (e.g., depending on the severity for the user), the Front-End provides the feature to define rules to handle this. Although the implementation of these rules requires a server-side component, the Front-End core provides the necessary user interfaces for such notification rules. A mock-up for this can be seen on the right side of Figure 14. The provision of server-side components for this is provided by the Configuration and Personalization building block described in below (see Section 4.1.3).

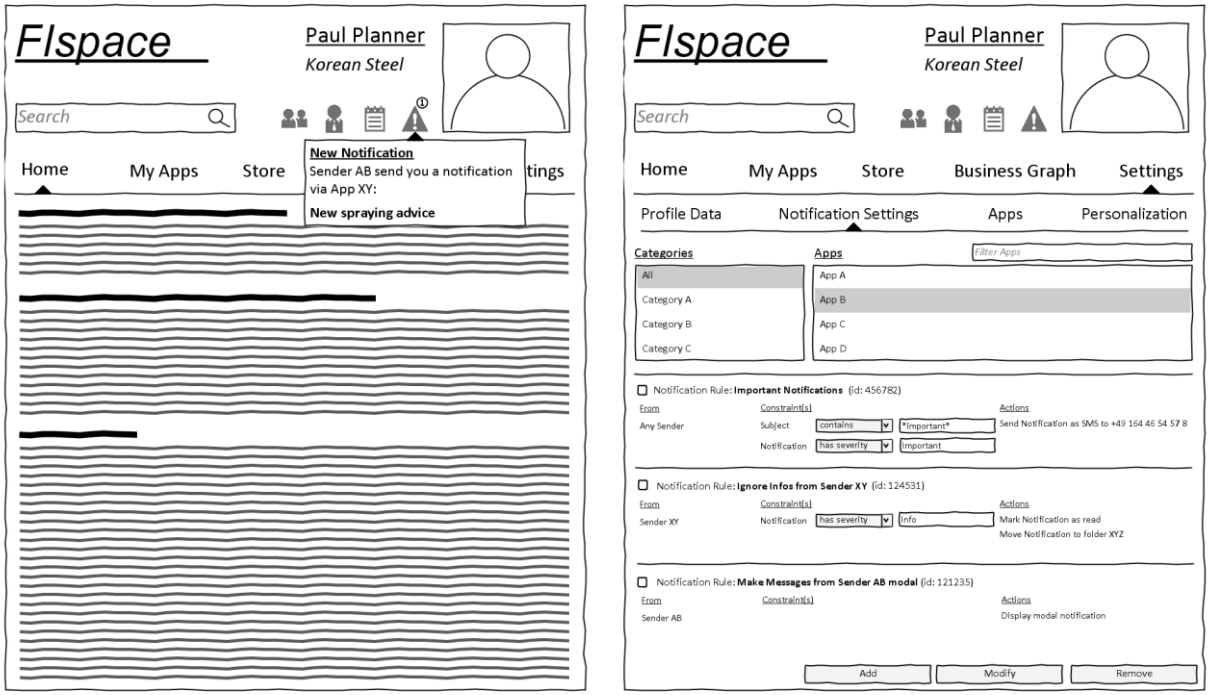


Figure 14: Reception of a new Notification and Management of Notification Rules

The Front-End Core is also responsible for defining user interfaces to adjust user and company profiles. Figure 15 shows a mock-up for this (user profile settings on the left side, company profile settings on the right side). We want to point out that the input fields used in the figure cannot be considered as complete. The definition of what data is necessary for person and company profiles is not fixed yet. Thus, the mock-up only gives a first impression of which information those profiles can encompass. A final decision only can be made in cooperation with the Use Case Trials. Also the server-side persistence and management of this information is not provided by the Front-End, but implemented by *Sub-Task 272 - User Management & Access Control*.

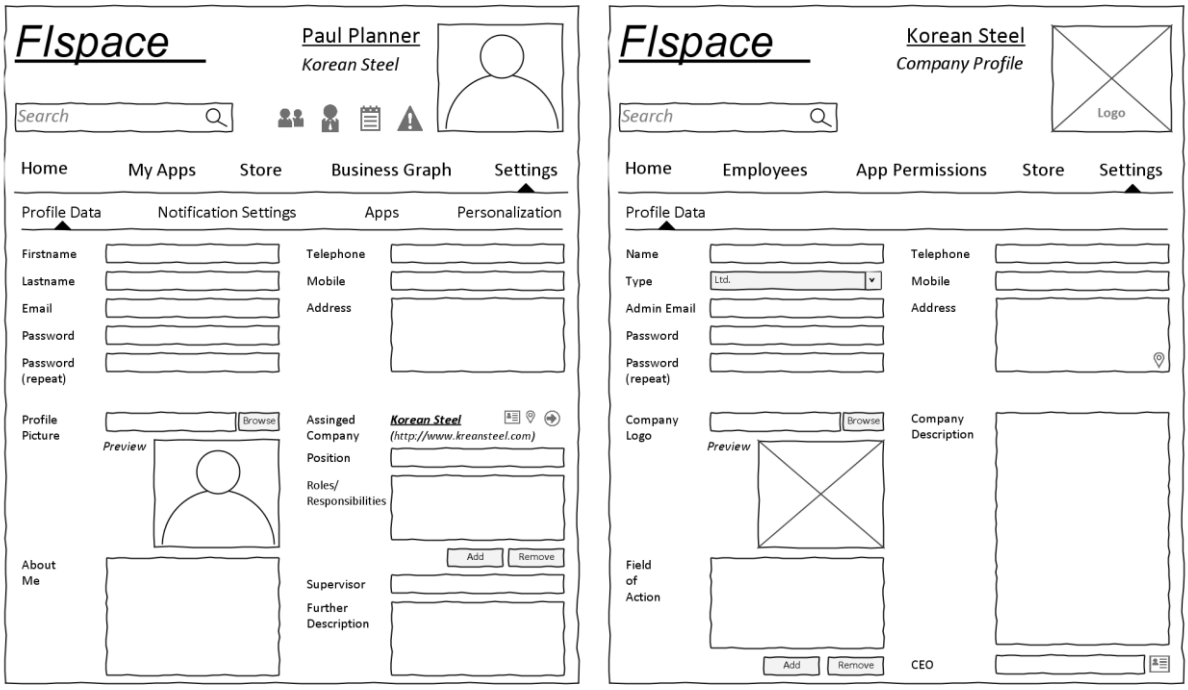


Figure 15: Profile Settings for User and Company Profiles

4.1.1.2 Planned Generic Enablers and Initial Technology Choice

In this section we present our initial assessment for Generic Enablers and potential other technologies for an implementation of the envisioned features of the End-User Core Front-End (Sub-Task T222). However, after assessing the current FIWARE GE catalogue, we came to the conclusion that there are currently no GEs provided which could contribute to the realization of End-User Core Front-End. For this reason we only present general frameworks and technologies.

The implementation of the End-User Core Front-End will encompass a client-side (user interface) and server-side (business logic) part. Please refer the High-level Technical Architecture in Section 4.1.6 for further details.

4.1.1.2.1 Potential Technologies for Client-side Components

As stated in the DoW and described in the sections above, the user interface will be implemented by a combination of HTML and JavaScript. We will make use of the latest developments on the HTML standard, widely known as HTML5. However, there will be no framework support necessary since most of the HTML code will generated during runtime by the JavaScript routines. Hence, most of the development effort will be done in JavaScript and to facilitate the development we intend to use the following frameworks:

- **SAP UI5** (http://scn.sap.com/community/developer-center/front-end)
  Modern UI framework to create web-application based on HTML(5) and JavaScript. It encompasses a widget library, a lightweight programming model based on the Model-View-Controller pattern and supports custom widget developments. This framework will build the foundation of the entire user interface and define the underlying project structure for HTML and JavaScript files.
- **jQuery** (http://jquery.com/)
  jQuery is very common library for DOM manipulation during runtime. It eases the handling of element selection, manipulation and events. Moreover, it provides bridges cross-browser-issues in many areas such as the AJAX. SAP UI5 is built upon jQuery and therewith, it can be used directly in SAP UI 5 applications.
- **CometD JavaScript Client** (http://cometd.org/)
  As mention in Section 4.1.6, HTTP is a request-/response-based protocol. Data is transferred from a server to a client only on requests from the latter. If a server wants to notify clients without a previous request, workarounds (client polling or long-polling based on the Bayeux protocol) or other protocols (WebSockets) have to be used. However, often clients and servers only support

specific options. The CometD framework abstracts from the concrete transportation methods and allow a negotiation between client and server to select the most appropriate technology.

4.1.1.2.2 Potential Technologies for Server-side Components

As described above, the UI Server provides service interfaces for the User Interface to access additional business logic that has to be provided on server-side. Additionally, the UI Server is responsible for delivering the User-Interface-related code artifacts to the clients. Hence, we need an implementation for server side logic, which also provides the necessary interfaces, and we need a web- or application-server which executes the implemented logic and is also delivers the User-Interface-related code artifacts to the client. Due to the fact that the FIspace will be most likely built upon Java platform, we focus our preliminary selection of potential technologies and framework on software executable on the Java Virtual Machine:

- **Apache Tomcat** (http://tomcat.apache.org/)
  Tomcat is an open source application server for Java-based applications. It implements the Java Servlet and Java ServerPages specification directly, but also can be extended to support the whole Enterprise Java Stack. We intend to use Tomcat as the main application server for implemented server logic and also to deliver User-Interface-related code artifacts to the client.
- **Eclipse Jetty** (http://www.eclipse.org/jetty/)
  Jetty is an alternative application server to Tomcat. It also implements the Servlet and ServerPages specification, but provides additionally support for the WebSockets protocol.
- **Spring** (http://www.springsource.org/)
  Spring is one of the most popular frameworks for enterprise Java applications. Originating from a dependency injection framework, it consists nowadays of a variety of different modules, which can be used in very different scenarios. Of special interest for the development of the UI Server are the following modules:
  - **Spring MVC:** Framework to design Model-View-Controller based web-applications. This module allows the easy creation of RESTful service interfaces. We intend to use Spring MVC as a binding component that links the service interface with the Java application
  - **Spring Security:** This module provides security extensions for web-applications. It allows session management and can be integrated with the Security, Privacy and Trust component of the FIspace.
  - **Spring Social:** Spring module that allows the connection of an application to other Software-as-a-Service provides as Twitter or Facebook. We could use this module to interconnect with other (social) web-sites if necessary.
- **CometD Java Server** (http://cometd.org/)
  The CometD framework allows the technology independent integration of server push mechanisms. This component is the server-side counterpart to the CometD JavaScript Client (s.a.).
- **Apache Camel** (Notification Service)

4.1.1.3 Release Plan

In this section we provide an initial release plan for the End-User Front-End Core. It is elaborated based upon our current understanding of the required features and our current effort estimation. Table 5 shows the current release plan.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Layout UI, Look&Feel	Define the overall layout of the FIspace Front-End Core UI and set-up basic look&feel	Initial	Refined	Final
User Registration	Implement registration for user profiles (necessary dialogs and server-side logic); also includes the involvement of the SPT component	Initial	Refined	Final
Company Registration	Implement registration for company profiles (necessary dialogs and server-side logic); also includes the involvement of the SPT component	-	Initial	Final
Login for User Profiles	Implement login (necessary dialogs and server side logic) for the user login; also includes the involvement of the SPT component	Initial	Refined	Final
Login for Company Profiles	Implement login (necessary dialogs and server side logic) for the company profile login; also includes the involvement of the SPT component	-	Initial	Final

User Profiles & their Manage-ment	Implement the basic UI for user profiles and their management (settings dialog)	Initial	Refined	Final
Company Profiles & their Manage-ment	Implement the basic UI for user profiles and their management (settings dialog)	-	Initial	Final
Notification UI	Implement the UI-side for App-based notifications; basically encompasses the display the hint for new notifications and provide UI elements to display them	-	Initial	Final
Home Screen	Provide a Home Screen for user and company pro-files with basic information and news	-	Initial	Final
App Integration UI	Provide the possibility to integrate FIspace Apps in the Front-End Core	Initial	Refined	Final
Store Integration	Provide the possibility to integrate the Store in the Front-End Core	-	Initial	Final

Table 5: Release Plan End-User Front-End Core

4.1.2 Integration of FIspace Apps

In this section we describe the main features and first results of Sub-Task ST223 – Integration of FIspace Apps. This encompasses a general overview, a description of the main features we elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for an implementation and a preliminary release plan for the coming three releases of the Front-End component. We address each of those aspects in a separate subsection below.

4.1.2.1 Overview & Main Features

This section addresses the envisaged functionality to integrate the UIs of FIspace apps within the Front-End Core as well as to allow the configuration and customization according to the user’s needs.

When selecting the item *My Apps* from the menu bar of the Front-End Core UI, the personalized app dashboard of the logged in user is displayed in the lower part of the screen as indicated in Figure 16. The rectangles in the lower part of the figure indicate different apps, which can have different sizes and are positioned according to the user’s preferences. When populating the personal dashboard, a user can select from all apps that are available in the store, provided that the app was purchased and the neces-sary access rights were granted to the user by the owner of the app. In case that the user’s profile is managed by the company, it could also be possible that company specific access restrictions apply to the selection of apps.

To organize his/her personal workspace a user can also create more than one dashboard and populate each one of them with a set of apps according to his/her needs and preferences e.g. create one dash-board with a group of apps that are handling a certain task/topic.

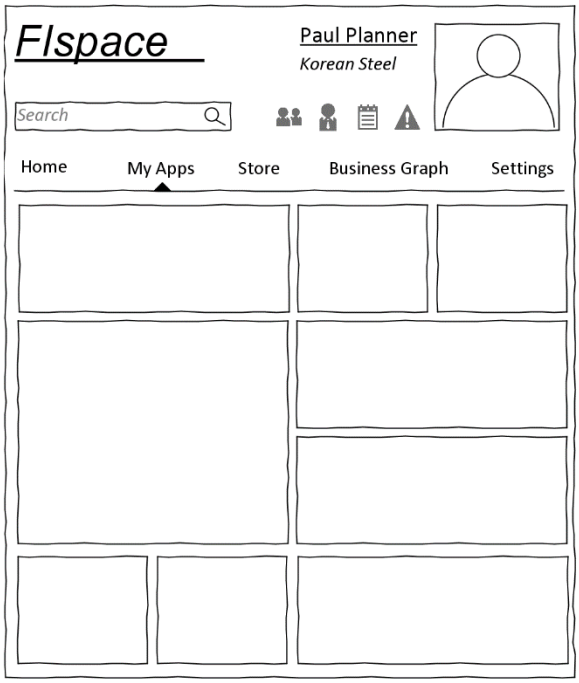


Figure 16: App integration dashboard

Besides selecting and positioning the apps according to his/her needs, the user shall also be able to define – as far as this is supported by the app – the size of each app on the dashboard. In case that the user needs more space for the usage of an app than it usually takes on the screen, the dashboard should also provide functionality to maximize the app.

Related to the connections between Front-End Core and apps there is also an option foreseen to link notifications from the Front-End Core to the related apps. When a user would read a notification like the one shown above in Figure 14, the UI should allow “bringing the app to the foreground”. The user should be able to click/select the notification from the Front-End Core’s notification/status bar. This “click event” would then “trigger” the dashboard to display the app that was the source (or is subject) of the notification. This could comprise bringing the correct dashboard to the front, moving the app to the top of the screen and possibly also maximize the app.

4.1.2.2 Potential Generic Enablers and Initial Technology Choice

In this section we present our initial assessment for Generic Enablers and potential other technologies for an implementation of the envisioned features of the Integration of FIspace Apps (Sub-Task ST223). We first discuss the WireCloud GE as a potential Generic Enabler which can be used for this Sub-Task and then present our preliminary technology choice.

4.1.2.2.1 WireCloud GE

WireCloud GE, as an open source, reference implementation of the FIWARE “Application Mash-up“ Generic Enabler, looks like a good basis for realizing the App Integration functionalities as described in sections 4.1.2.1. It already provides the following functionalities:

- Interface to an App Store
- Management of user dashboards
- Creation of mash-ups of apps through so-called “wiring” (see section 4.1.2.2.2)

Several adaptations and enhancements would have to be implemented for WireCloud within the FIspace project to fully achieve the envisaged functionalities:

- Integration of FIspace SPT and User Management Services instead of WireCloud’s own user management
- Integration of App Invocation Services (see section 4.1.6.3) to invoke apps from the Front-End Core

- Integration of Configuration and Personalization (see section 4.1.3) features for dynamic customization of UI

4.1.2.2.2 Initial Technology Choice

As described in section 4.1.2.1, the App Integration feature provides functionalities to organize and customize a user’s personal dashboard to display the UI parts of different FIspace Apps. There are several tools available, which allow for this kind of dashboard management. The following presents a non-exhaustive list of such tools, which have been considered to be used for the FIspace project.

- **WireCloud GE** (http://conwet.fi.upm.es/wirecloud/)
  As already described in section 4.1.2.2.1, Wirecloud is a reference implementation of the FI-WARE “Application Mashup“ Generic Enabler. It provides a mashup engine based on the W3C Widget specification[6]. In addition to functionalities for personalized dashboards, WireCloud is able to interface with different app stores, from which apps can be obtained. It also comes with its own graphical wiring editor that allows for creating mashup applications out of two or more apps by connecting the output parameters of one app with input parameters of another app.
- **Apache Rave** (http://rave.apache.org/)
  Rave is an open source social mashup engine developed under the umbrella of the Apache Software Foundation. It integrates the Apache projects Shindig[7] and Wookie[8] to enable hosting of OpenSocial[9] and W3C Widget compliant applications. It includes a basic app store for publishing apps as well as a full-fledged user management. Apps published in the app store can be added to and freely arranged on a user’s personal workspace.
- **JBoss GateIn** (http://www.jboss.org/gatein)
  GateIn is one of several enterprise portals based on the Java Portlet Specification[10]. Like Wirecloud and Rave, GateIn allows a user to create his/her personal dashboard and populate it with several apps (so-called portlets). It also offers an integrated app store and a comprehensive user-management.

4.1.2.3 Release Plan

In this section we provide an initial release plan for the Integration of FIspace Apps. It is elaborated based upon our current understanding of the required features and our current effort estimation.The table below shows the current release plan.

Feature	**Description**	**V1** (M9/12)	**V2** (M15/18)	**V3** (M21/24)
App Dashboard Personalization	Positioning and Re-sizing apps on an App Dashboard Manage multiple App Dashboards (create/edit/remove)	Initial	Refined	Final
App Store Connection	"Download" Apps from the App Store and add them to a personal App Dashboard	Initial	Refined	Final
App Invocation from Front-End Core	Focus an App on an App Dashboard when clicking a notification in the Front-End Core	-	Initial	Final
Integration with FIspace SPT/User Management	Associate App Dashboard(s) to a user and store the user’s App Dashboard settings/preferences in SPT	-	Initial	Final
Integration with FIspace SPT/Authorization	Filter accessible Apps based on access rights granted to a specific user	-	Initial	Final

Table 6: Release Plan Integration of FIspace Apps

[6] http://www.w3.org/TR/widgets/
[7] http://shindig.apache.org/
[8] http://wookie.apache.org/
[9] http://opensocial.org/
[10] http://www.jcp.org/en/jsr/detail?id=286

4.1.3 Personalization and Configuration for End-Users

In this section we describe the main features and first results of Sub-Task ST224 – Personalization and Configuration for End-Users. This encompasses a general overview, a description of the main features we elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for an implementation and a preliminary release plan for the coming three releases of the Personalization and Configuration for End-Users component. We address each of those aspects in a separate subsection below.

4.1.3.1 Overview & Main Features

The FIspace platform shall support the configuration of “customized end user cockpits”, i.e. provide functionality for personalizing and configuring the Front-End for the individual needs of users. Besides the UI adaption by users according to their personal preferences, another possible scenario could be a company configuring the UI for their users’ accounts, adapting the UI to the users’ tasks and adjusting it to follow corporate design guidelines.

Some of the envisaged options for configuration and personalization were already addressed in previous subsections, namely the notification settings – i.e. enable the user to define about which events or messages he/she wants to be informed over which way and channel (see section 4.1.1.1 and especially Figure 14) – and the personalization of the app dashboard (see section 4.1.2.1).

Additional configuration options to be supported are the personalized/corporate design of the FIspace Front-End and the option to enable or disable specific features of FIspace (e.g. certain UI parts or access to the App Store). The option to enable only specific features could be relevant for both companies (restricting their users access rights to just the features they need) and end users (disabling non-needed features to reduce complexity and to avoid errors).

In general the configuration and personalization options relevant for the overall Front-End should be reachable via the *Settings* item of the Front-End Core’s menu bar (e.g. via a sub-menu as indicated above in Figure 15). Exceptions of this rule could make sense for options that apply only to a specific part of the UI, e.g. the positioning of apps on the dashboard.

4.1.3.2 Potential Generic Enablers and Initial Technology Choice

An implementation of the envisioned features for the Personalization and Configuration for End-Users is by its nature tightly bound to the End-user Front-End Core (see section 4.1.1) and Integration of FIspace Apps (see section 4.1.2) development. Hence, the used implementation technologies have to be aligned. Consequently, technology choices for the Personalization and Configuration for End-Users are dependent on and will follow the choices made for the End-user Front-End Core and FIspace App Integration.

4.1.3.3 Release Plan

In this section we provide an initial release plan for the Personalization and Configuration for End-Users. It is elaborated based upon our current understanding of the required features and our current effort estimation. The table below shows the current release plan.

Feature	**Description**	**V1** (M9/12)	**V2** (M15/18)	**V3** (M21/24)
Personalized UI settings	Adapt UI look and feel according to user’s preferences and/or company’s corporate design rules	-	Initial	Final
FIspace Feature Selection	Enable/disable FIspace features based on user’s preferences and/or company’s policies	-	Initial	Final
Notification settings	Enable the user to define about which events or messages he/she wants to be informed over which way and channel	-	Initial	Final

Table 7: Release Plan Personalization and Configuration for End-Users

4.1.4 Social Networking & Collaboration Features for Business Communities

In this section we describe the main features and first results of Sub-Task T225 –Social Networking & Collaboration Features. This encompasses a general overview, a description of the main features we elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for an implementation and a preliminary release plan for the coming three releases of the Social Networking & Collaboration Features. We address each of those aspects in a separated subsection below.

4.1.4.1 Overview & Main Features

The DoW denotes the Social Networking & Collaboration Features as a set of tools for communication, information exchange and networking. By this, the collaboration between different partners shall be facilitated in order to ease the implementation of business transaction. The term set indicates that this component consist of more than just one feature, but provided different sub-parts which are supposed to be integrated at different ‘places’ of the Front-End UI. Currently, we have planning to provide three different sub-parts:

1. Business Partner Management
2. Facilitated Communication on Business Transactions
3. Business Network Analytics

We discuss each sub-part in a separate section below.

4.1.4.1.1 Business Partner Management

The Business Partner Management allows users to add business partners to their personal contact list. This is comparable to an address book and the concept is also employed by the majority of current social networks (e.g., Facebook friends, or LinkedIn connections). By also adapting this concept for the FIspace we want to give the user easy access to their regular business partner and therewith facilitate the business management. In contrast to a traditional address book the network data is managed by the FIspcae in a central manner and the user does not have to update the contained information. Users only add a link to a preferred business partner and with that are able to always have access to the most recent information about this partner.

Although we consider the Business Partner Management as a substantial part of the Social Networking & Collaboration Features it is also addressed by the Baseline App Business Service & Contract Management as presented in Deliverable D400.1. We are currently discussing how we can combine our efforts and deliver a holistic business partner management to the user. More details about this will be provided in Deliverable D200.2 – Technical Specification.

4.1.4.1.2 Facilitated Communication on Business Transactions

Businesses often rely on transaction between (distinct) parties. Depending on the complexity of a transaction more or less intensive communication between the parties is required. In order to ease the communication between different parties as much as possible we want to enable users to start communicating with each other directly under the FIspace representation of such a business transaction. Modern social networks provide a comparable feature where users can directly comment a posting or event of other users. We want to apply this concept for business transactions in order to facilitate the communication between business partners as much as possible. Furthermore, it shall be possible to send messages directly to business partners and other FIspace users and start also a real-time chat.

Currently we are checking the requirements for such a feature. So far there is no entity in the FIspace platform to such a business transaction. However, we consider this as a basic requirement to implement such a communication features. We are currently discussing with the App Integration team as well as the development team for Baseline Apps how this can be supported.

4.1.4.1.3 Business Network Analytics

Life is full of opportunities, but stumbling on them is often a matter or luck. Just imagine how often you have heard somebody say “If only we had known about this person / this project / this company etc. earlier”. What if we could automatically alert you about important facts about the people and companies around you and help you discover new opportunities for your business?

Social networks have existed before Facebook but the relations between friends were stored in many different systems, like email, IM messenger, address book, etc. Likewise, business networks already exist today but the information is stored in many different systems: customer relation information in the CRM system, supplier information in the SRM system, your personal communication in your email, your business contacts in LinkedIn, etc. Just like Facebook has revolutionized the way people communicate by integrating your personal communication means in a single platform, the social business network intelligence graph aims to revolutionize the way companies communicate with each other and their customers.

As mentioned, the Front-End Core provides features to manage business relationships. The *Social Business Network Intelligence Graph* (SBNI-Graph) analyses those relationships and make the resulting information visible to the user and facilitates the understanding of the business network indicating pain point of the user’s business network. The SBNI-Graph owns two parts, the analytics part, which analyzes the available network data and second the graph visualization, which visualizes the information from the analytics part. Apart from business partner relations, we currently investigate other sources for the business network analytics. Transaction data between suppliers and customers could be an example for this. However, we are fully aware that this would be sensible data, which can only be provided for distinct users. Using these data will generate interesting questions and answers using this graph:

- Cluster analyses
  - Which are the most valuable nodes, clusters, connection within your network?
- Predictive analyses
  - What will happen if you change something today?
- Change awareness
  - Who is involved in what – which are critical dependencies?
- Exploration analyses
  - Identification of secret relations and interdependencies that wouldn’t be visible today

The related data are about objects and their relations. Therefore a graph is used as visual representation. The target of the SBNI-Graph is to get a more feasible representation of your business network using a graphical visualization. Graphical visualization of data is easier to percept instead of long tables, list and matrices.
Graphs offer an exploration experience of large information. Graphs can contain sub graphs that could be visualized in a different level of detail. This allows the user to explore the graph and build up his own understanding of the network. Collapsing does exploration and expanding nodes, zoom, move objects to different places (keeping the relation) or panning the stage.
But never the less graph answer many question (you even not asked yet) like question about affinity, information flow, central nodes and active relations, visualizing subnets and single players.

These data provide rich information for analytics to identify the state of the art of the business network. The goal of the Social Business Network Intelligence Graph is to build novel search and analytics tools for enterprise software systems. The key ingredient of the solution is a flexible graph data structure that represents the relations between people, companies and other entities in the business from various structured and unstructured data sources. By explicitly representing the data as a graph, we believe that we can “connect the dots” between different systems and discover previously unknown synergies and opportunities.

4.1.4.2 Potential Generic Enablers and Initial Technology Choice

In this section we present our initial assessment for Generic Enablers and potential other technologies for an implementation of the envisioned features of the Social Networking & Collaboration Features (Sub-Task T225). However, after assessing the current FIWARE GE catalogue, we came to the conclusion that there are currently no GEs provided which could contribute to the realization of End-User Core Front-End. For this reason we only present general frameworks and technologies. The following provides a list of technologies we took in consideration so far:

- **D3 (Data-Driven Documents)** (http://d3js.org/)
  D3 is a library that allows the creation of different data visualizations based on data. It provides a variety of predefined graphs and other diagram types which can be used to make information more comprehensible for the user. We assume that this library especially will be used for the visualization of business relationships.
- **Neo4j** (http://www.neo4j.org/)
  Neo4j is a graph-based database which can be used to represent the relationships between users and companies. However, the storage of user data is covered by the Security, Privacy and

Trust component of the FIspace platform. For this reason we assume that we do not have to provide an own database to persists user and transaction data, which is needed for the visualization.

4.1.4.3 Release Plan

In this section we provide an initial release plan for Social Networking & Collaboration Features for Business Communities. It is elaborated based upon our current understanding of the required features and our current effort estimation. Table 8 shows the current release plan.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Business Partner Management	This feature represents the basic business partner management and it enables users to add business partners or preferred companies. However, was we wrote in Section 4.1.4.1.1 there are some crosscutting concerns with the Business Service & Contract Management Baseline App and we have to determine which part has to be implementing by which group. For this reason we put the releases for the milestones in brackets to mark them as optional.	-	(Initial)	(Final)
Facilitated Communication on Business Transactions	This feature encompasses the facilitated communication based on business transactions. Due to the fact that we depend on the development of business transaction in the FIspace platform we cannot provide an initial release until the V1 milestone.	-	Initial	Final
Business Network Analytics	This feature encompasses the visualization of business relationships based on business related data contained in the FIspace platform (cf. Section 4.1.4.1.3 for further details about the possible type of transactions).	-	Initial	Final

Table 8: Release Plan Social Networking & Collaboration Features for Business Communities

4.1.5 Ubiquitous Access

In this section we describe the main features and first results of Sub-Task T226 – Ubiquitous Access. This encompasses a general overview, a description of the main features we elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for an implementation and a preliminary release plan for the coming three releases of the Ubiquitous Access component. We address each of those aspects in a separated subsection below.

4.1.5.1 Overview & Main Features

In this section we describe the basic features of the Ubiquitous Access (UA) component of the Front-End. The main functionality of this component is the transformation of the user interface according to the user’s device capabilities, as well as information from the network that could be provided from the client side, i.e. latency, bandwidth, etc. The users shall have no direct interaction with the UA component. The UA component shall be acting in the back-end, as an external component to the UI Core component, with which will be exchanging the needed information to provide the final adapted content (when needed) to the user.

To accomplish this, UA component has to publish APIs that will be used from the UI Core component to send to the UA the necessary user information (device specifications, network information etc.). After receiving the required information, UA component shall take care of adapting the content and the appearance of the UI according to pre-specified rules, which will comprise the Adaptation Mechanism - Logic.

As a result, the user will be experiencing an adapted to his/her own device and network characteristics User Interface.

4.1.5.2 Potential Generic Enablers and Initial Technology Choice

In this section we present our initial assessment for Generic Enablers and potential other technologies for an implementation of the envisioned features of the Ubiquitous Access adapter (Sub-Task T226). How-

ever, after assessing the current FIWARE GE catalogue, we found two GEs that could provide some benefits for this sub-task. However, they are not implemented yet and it is hard to give a more detailed assessment.

4.1.5.2.1 Connected Device Interfacing GE

Connected Device Interfacing (CDI) Generic Enabler (GE) is part of the Interface to Networks and Devices (I2ND) family of GEs. CDI will be implemented as a JavaScript library and is a set of common JavaScript APIs which exists on a range of platforms, allowing developers to create web applications. The following is a short description of the CDI functionality:

1. Device Sensors (e.g. camera, microphone, Geolocation, Device Orientation , Sensor Discovery)
2. Quality of Experience (e.g. start/stop monitoring, Give QoE feedback)
3. User Profile (e.g. Access to user profile information, Access to local and personal device data)
4. Device Features(.g. Screen size, processor type, connectivity, battery state)
5. Personal Data Services(e.g. File System Access, Contacts, Calendar)

Taking into consideration that the CDI can be deployed on a wide range of devices (iOS, Android, PC platforms, etc.) according to the FI-WARE latest reports, the utility of the particular GE becomes clear. It has to be mentioned, however, that until the time of this writing, no implementation has been provided.

4.1.5.2.2 Network Information & Control GE

Network Information and Control (NetIC) GE is part of the I2ND family of GEs as well. Its main functionality according to the FIWARE shall be:

1. Network Resource Control (e.g. activate/deactivate network interfaces, manage physical infrastructure)
2. Network Statistics - This functional block will offer more detailed information about the status of links and paths
3. OpenFlow Network Information && Control - This functional block will offer detailed information about the status of an OpenFlow network, and also control capabilities to manage it
4. Topology OPT Location - The client provides the network with its own address and a list of potential source addresses. The network internally analyzes the load situation and returns the best suited source address

The NetIC functionality also fits perfectly for the needs of the content adaptation task. Especially, the Network Statistics functional block could provide the largest part of the information required from the Ubiquitous access component. This GE, though, is not implemented as well, according to the latest FI-WARE reports.

4.1.5.2.3 Initial Technology Choice

The Ubiquitous Access (UA) component will also build in Java Virtual machine in compliance with the rest components. A vital part of the UA component is the communication with the UI server in order to publish and receive the necessary data. Thus, potential technologies for the particular functionalities of the UA component could be:

- **Apache Tomcat** (already described in the section above – UI Server)
- **Grizzly** (https://grizzly.java.net/)
  Grizzly is an open source HTTP protocol framework for creating custom HTTP applications and a HTTP Server framework which offers high level abstractions to the HTTP protocol
- **Jersey** (https://jersey.java.net/)
  Jersey is also an open source framework. Jersey implements support for the annotations defined in JSR-311, making it easy for developers to build RESTful web services with Java and the Java JVM.

4.1.5.3 Release Plan

In this section we provide an initial release plan for Ubiquitous Access component. It is elaborated based upon our current understanding of the required features and our current effort estimation. The Ubiquitous Access component is directly dependent to the Front-End Core component, as the Core will be the ‘con-

tent provider’ for the adaptation mechanism. Thus, the release plans of the two components are tightly related. Table 9 shows the current release plan.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Http Receiver	Implement the component, which will receive some basic information with regard to the client’s type of users agent.	Initial	Refined	Final
User-Profile Receiver	Implement the component, which shall check the type of the profile (simple users / company) and will provide this info to the Aggregator		Initial	Final
Hardware Receiver	Implement the component, which will receive the client’s device capabilities an specifications (screen size, battery, sensors etc.)	Initial	Refined	Final
Network Receiver	Implement the component, which will receive and provide to the Aggregator, network information of the user (e.g. latency, throughput etc.)		Initial	Final
Initial CSS, HTML, JS Receiver	Implement the component, which will receive the initial content from the Front-End Core and will provide it to the Adapter Processor for the adaptation process	Initial	Refined	Final
Aggregator	Combine all the information available from the “Receiver” components to provide it to the Adapter Processor	Initial	Refined	Final
Adapter Processor	Implement the component, in which all the adaptation rules are defined and executed for transforming the initial content to the adapted one to be forwarded back to core	Initial	Refined	Final

Table 9: Release Plan Ubiquitous Access

4.1.6 High-level Technical Architecture

After the discussion of the main features of the Front-End we now present and describe the conceptual architecture of this component. Figure 17 shows a graphical overview of the Front-End using a Component Diagram of the Unified Modeling Language (UML). As indicated in the figure, the conceptual architecture of the Front-End consists of four main subcomponents: *User Interface, App Integration Server, UI Server* and *Ubiquitous Access Adapter*. Those are interconnected to each other and also hold connections to other FIspace components, such as the B2B Collaboration Core or the Security, Privacy & Trust component. In the remainder of this section we describe each subcomponent in a separated subsection.

4.1.6.1 User Interface

The User Interface subcomponent represents all parts that are directly visible for end-users. As stated in the Description of Work (DoW), the visible interface for end-users is foreseen to be web-based and shall be realized by a combination of HTML, CSS and JavaScript code artifacts. Those artifacts are interpreted by a web browser during the interaction of the end user with the interface. The browser interprets the code artifacts in order to transform the declarative (HTML, CSS) and imperative (JavaScript) interface descriptions into a set of visual representations and reactions to user inputs. Hence, the User Interface subcomponent encompasses all code artifacts that are intended to be interpreted on client-side.

The User Interface also contains several internal modules. It is important to note that those modules focus on the creation of visual user interfaces. Some of the modules are also replicated in other subcomponents of the Front-End, whereas these provide server-side logic. Only through the combination from user interface and service-side service end-users will be able to use the foreseen features of the FIspace. The *Core UI* module encompasses HTML, CSS and JavaScript artifacts, which are used to implement the features described as the Front-End Core in Section 4.1.1. The Business Network Analysis UI provides the user interface to visualize the user’s business network relations as described above. The App Dashboard UI Integration and Store UI Integration target at the integration of the externally provided UIs of the App Dashboard and FIspace store.

This subcomponent implements mainly functionalities closely related to the user interface. Hence, it is not foreseen to implement advanced business logic. To enable access to such functionalities the User Interface uses services provided by the other subcomponents of the Front-End. This is indicated in the diagram via the *‘access front-end services’* channel. Another channel is labeled *‘app invocation’* and targeted at the App Integration Server subcomponent. Via it the User Interfaces communicated with the App Inte-

gration Server about App-related user actions in the core Front-End. With this, for example, the App Integration Server is informed about a click on a notification of a specific App. The App Integration Server can uses this information to bring this App to the foreground.

4.1.6.2 UI Server

The UI Server provides the necessary server-side services for the User Interface and other subcomponents. It also hosts the code artifacts of the User Interface and delivers them to a requesting client. The delivery will be handled by the HTTP protocol, which also ensures an easy access with web browsers. However, HTTP is stateless and unidirectional. Due to the fact that the Front-End has to provide user interfaces for a variety of different user at the same time, a stateless protocol is not sufficient. Modern web-based applications solve this issue by the implementation of client sessions. The UI Server holds the necessary logic for this and enables the stateful handling of clients. For this, it introduced a User Mgt. Service, which is also being used by other modules or subcomponents. This service connects to the Security, Privacy & Trust component to use the provided features of its user management.

The Notification Service module represents the server-side element to send notifications to the User Interface. As described above, the User Interface is executed on client-side and thus, only in operation if a user is connected. In order to process and deliver notifications even to users who are not currently connected a server-side element is needed. The notification service used functionalities of the User Mgt. Service in order to receive user-related information. Due to the limitations of HTTP (as mentioned, it is unidirectional) a server is not able to push information to the client directly. For this, the Push Service provides the additional logic to enable a server push via Bayeux or Web Sockets for example.

The Business Network Analysis Mgt. provides the server-side element for the Business Network Analysis UI and is mainly concerned with the aggregation of the right data. The Settings Management Module stores user-specific UI settings and loads them if the user reconnects. Thus, it also uses the User Mgt. Service as a facade to access the Security, Privacy & Trust component. The Ubiquitous Access Integrator is also connected with the Settings Management Module in order to take the specifics of the user’s device into account.

4.1.6.3 App Integration Server

The main task of the App Integration Server is to render the dashboard including the apps, taking into account the user access rights and overall design settings as well as user specific dashboard settings. The App Integration Server comprises four sub-components: App UI Hosting, App Dashboard Service, App Invocation Service and App Dashboard Settings Service.

The main task of the App UI Hosting component is to host the UI part of the FIspace apps that are available in the FIspace App Store. For this, the App UI Hosting component offers an interface to the FIspace store for deploying App UIs, which will be based on HTML, JavaScript and CSS. The App UIs are envisaged to comprise only a minimal amount of business logic, while the rest of the logic is provided by the app’s backend. This exchange of business data with the backend is indicated in Figure 17 by the connection between the App UI and the B2B Collaboration Core.

The set of available apps is, in general, equivalent to all apps that are deployed to the App UI Hosting component by the FIspace store, but the set of apps accessible for a specific user can be further restricted by user access rights to be provided by the SPT module.

The App Dashboard Service component provides (renders) the user-specific dashboards to the User Interface component and enables the user to configure the dashboards according to his/her needs (e.g. select the apps to be displayed as well as adjust their positioning and size).

The App Invocation Service component allows the User Interface component to interact with the apps on a user’s dashboards. To allow for the invocation of a specific app by the User Interface component (e.g. when a user clicks on an app-specific notification in the User Interface) a channel from the UI to the App Invocation Service component is provided.

To pass UI settings to the App Dashboard a so-called App Dashboard Settings Service is provided, where the core UI can pass those settings to the App Integration Server that do not just apply to the Front-End Core but to the whole GUI (e.g. change of colors, corporate design, look & feel).

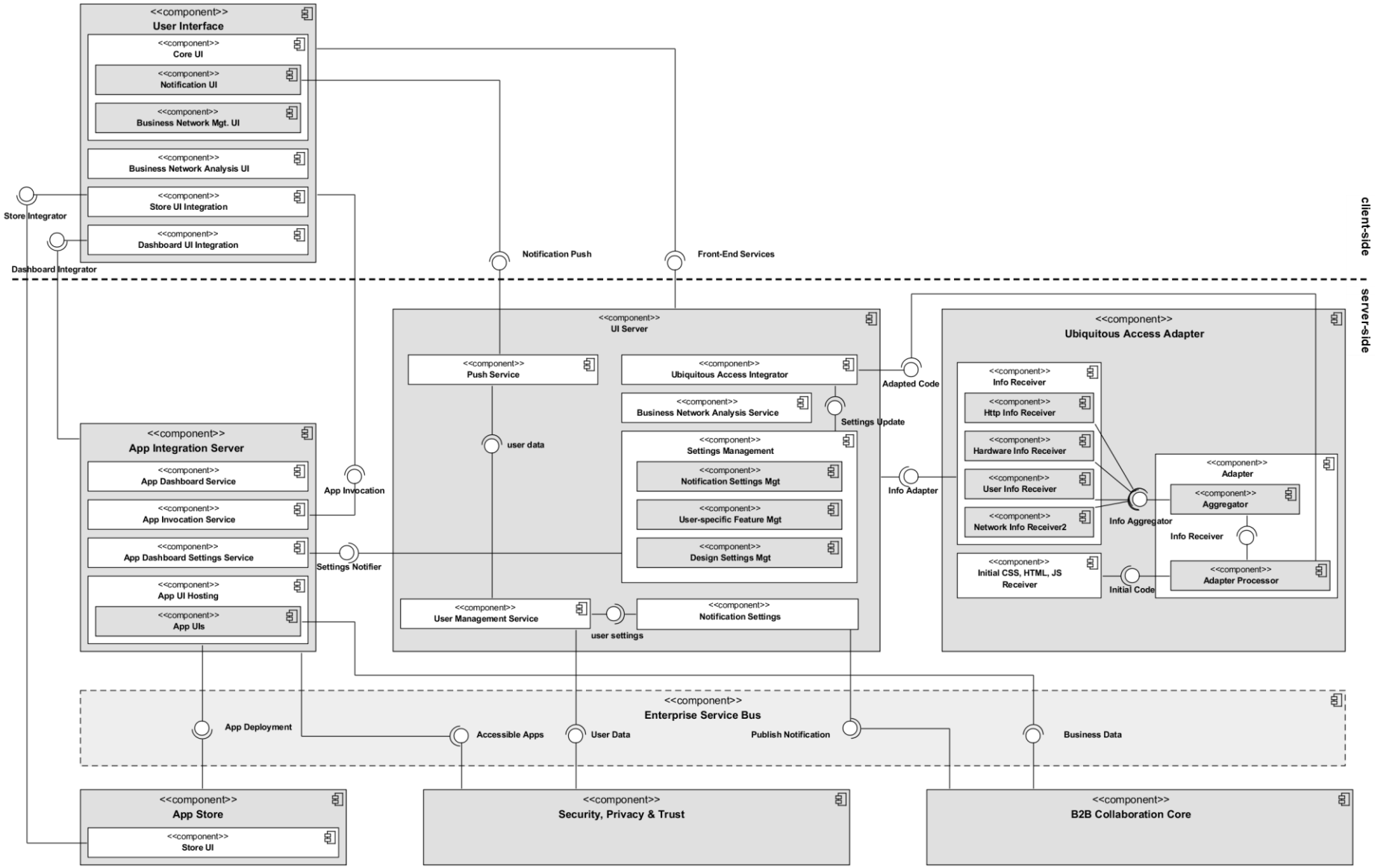


Figure 17: Front-End Conceptual Architecture

4.1.6.4 Ubiquitous Access Adapter

The Ubiquitous Access (UA) component –as depicted in the conceptual architecture above (Fig. 7)- consists of three main sub-components and sub-entities:

- The Info Receiver, which also consists of:
  - The HTTPInfoReceiver
  - The HardwareInfoReceiver
  - The UserInfoReceiver
  - The NetworkInfoReceiver
- The Initial CSS/HTML/JS Receiver
- The (Main) Adapter component, which takes care of aggregating the collected info and applying to it the adaptation rules. For these two sub-functionalities, the Adapter consists of two respective sub-components:
  - The Aggregator
  - The Adaptation Processor

Ubiquitous Access Adapter shall publish interfaces that will be used from the UI Core component to gather profile, device and network information from the user. The Info Receiver entity will be responsible for the reception of the necessary information. Upon reception, the information is forwarded to the Adapter entity, which –as already mentioned- shall aggregate the information (Aggregator component) and apply the rules (Adaptation Processor) that will finally result in the adapted content, ready to be forwarded back to the UI Core.

4.2 The FIspace Store

The FIspace Store is concerned with the software infrastructure to allow for the provisioning and consumptions of FIspace Apps, therewith providing the core elements for the monetization throughout the ecosystem that shall be facilitated by the FIspace. All FIspace Apps shall be made available in the Store and consumer will be supported with easy to use search and consumption features. The consumption includes the purchase support as well as the slight execution at runtime. Features for the former contain an App purchase process. Features for the latter include easy download and installation (if components need to be installed on client side) as well as unlocking (for components running directly in the cloud) and execution of Apps and components. Finally, for consumers also a simple right management must be provided which informs the customer about mandatory and optional rights the App requires (before purchase) and enables him or her to configure those for each App (after purchase). For App publishers and developers supportive features will be provided too. Firstly, easy to use discovery for Apps and App interfaces are intended to enable them to find reusable functionalities. Secondly, Publication support is provided together with an integrated compliance check for publishing new Apps in a simple way in the FIspace store. Finally, financial management is part of the FIspace Store which enables app providers to run statistics and share revenue with involved partners (e.g. developer of re-used component) using different revenue models.

The remainder of this section is structured as follows: In the next section we give a general overview of the FIspace Store. Afterwards, we define its main features and usage processes along with the initial development plan. Finally, we present the results of our current investigations of potential Generic Enablers that are considered to be used for the technical implementation.

4.2.1 Overview

In general all Apps are stored in a so called FIspace App repository. As shown in the figure below, the FIspace Store has 2 main user groups: Consumers as the (Business) End-Users who utilize the FIspace apps to conduct business tasks (esp. the collaborative business activities across organizational boundaries), and Developers who develop Apps for the FIspace.

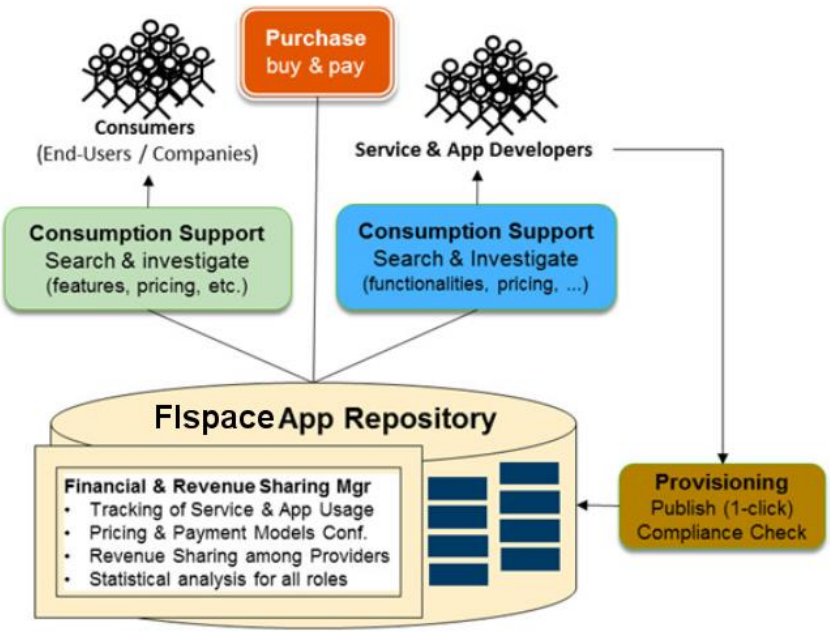


Figure 18: FIspace Store Overview

For both, a Consumption Support component is provided which provide the necessary software infrastructure to find, and (re-)use Apps, resp. provide them via the FIspace Store. Apart from that, a Purchase component will be implemented which supports the purchasing process including the process support for buying an App (e.g. incl. terms and condition agreement) and paying for it. In the figure below you can also find a Provisioning Support component which enables developers to easily upload and publish

newly developed Apps (e.g. including technical compliance checks). In addition, the ‘Financial & Revenue Sharing Manager’, who is part of the already mentioned FIspace App Repository, handles the economic aspects, i.e. the monetary incomes and the revenue sharing to the providers’ ecosystem.

Technically, the FIspace Store and the necessary software infrastructure is planned to be developed on top of existing frameworks and technologies; primary candidates are the Generic Enablers from the FI-WARE IoS Chapter, esp.: using (Linked) USDL as the basis for the App Description Language, using the Service Repository as basis for the FIspace App repository and the Marketplace GE as basis for features of the Consumption Support and Provisioning components.

Based on the initially developed concept of the FIspace Store and its gross-grained feature description we elaborated a first paper-based mock-up in order to bring everything together. A more precise description of the designated features of the Store can be found in the next three subsections. Our mock-up, shown in Figure 19, in segmented in different areas, whereas each implements a specific part of the store: On the top of the mock-up a combo-box and a text field can be found. Both enable the search for provided Apps; the first enables the user to select and browse categories and the latter provide access to a free text search. Blow that, the logo, the description and the rating of a specific App is provided (assuming the user has already selected one). This part also contains a purchase button to directly buy the App. In the next area, further details about the App and its features are provided. This includes also additional advertisement material, such as screenshots or videos. Only indicated in the mock-up is the access to additional material, the User Guide, Technical Information, detailed Reviews and License Information.

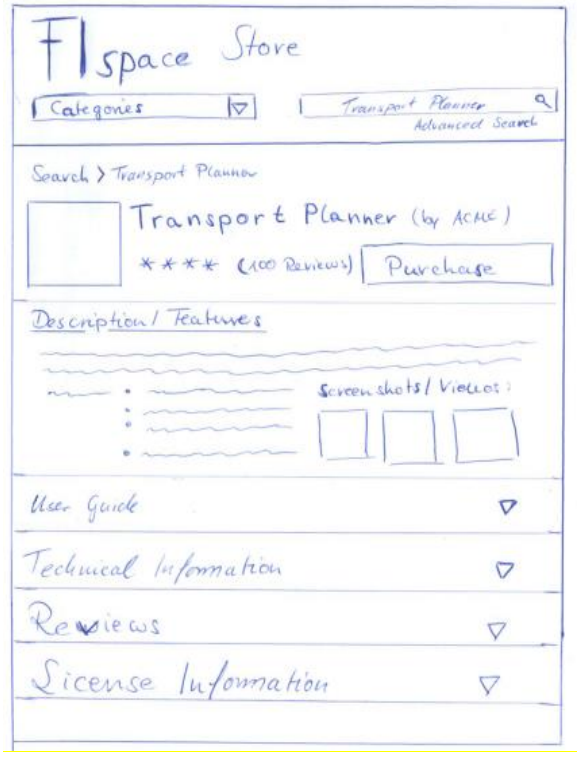


Figure 19: Paper-based Mock-up for the FIspace Store

4.2.2 App Repository & Provisioning Support Features

The App Repository & Provisioning Support process encompasses all steps a developer has to do in order to upload an App to the Store and make it available to End-Users. In general three different aspects can be identified:

1. **Create App Description**
   Based on Linked USDL the developer has to describe the provided services of his/her App. Linked USDL is provided by FIWARE and enables the domain independent description of provided services. For this, a specialized editor can be use which is provided by the Linked USDL development team from FIWARE (see GE validation below). The SDK component currently investigates the possibility of integrating it in the FIspace SDK (*cf.* Section 4.7)
2. **Define Usage and Pricing Model**

The next step would be that the App developer picks a pricing model that he/she wants to apply for the App. For this the Store development teams defines a predefined list, which might be extend by the App developer. Additionally, information about the foreseen usage model and the software license has to be provided by the developer.

3. **Upload App and Provide Documentation**
   The last step a developer has to do if he/she wants to provide an App over the FIspace store is the upload of the created code artifacts (e.g. App bundle) to the store. With this the store might also allow to upload additional resources as advertisement pictures or videos, documentation as well as re-use information for developers.

Under consideration of these different aspects we were able to identify a set of features the App Repository & Provisioning Support process contributes to the Store. Table 10 summarizes this.

Component	Description	Relevant FIWARE GEs
FIspace App Repository	**The repository where all registered FIspace Apps are available (most likely based on (Linked) USDL descriptions), including**: • Storage of FIspace Apps, resp. their linked USDL descriptions • Extension of (Linked) USDL for description of Apps • USDL Editor to create descriptions for FIspace Apps • Basic CRUD (create, remove, update, delete) operations for managing Apps	• Apps Registry • Apps Repository • Linked USDL GE

Provisioning Support for Developers	**Tool Support to allow Developers to publish Apps in the FIspace Store, incl.:**
	• Provisioning procedure:
	  ○ register as developer / provider
	  ○ create and publish USDL description for App
	• ‘Technical Governance Check’: support for FIspace Technologies (Interfaces, UIs, APIs, Security), functional check
	• Define / configure pricing model and usage terms & conditions, etc.
		• Apps.Repository
		• Marketplace GE
		• Linked USDL GE

Table 10: Necessary Features for the App Repository & Provisioning Support Process

Based on this feature definition we are able to provide an initial release plan. The table below enlists the features together with an overview of the expected milestones.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
App Repository	Repository / Registry for keep the meta-data description of FIspace Apps	Initial	Refined	Final
Upload support to Store	Tool support for App Developers to upload a FIspace App to the Store	Initial	Refined	Final
Integration with SDK	Integrating the App provisioning support (Store) with the App Development Environment	Initial	Refined	Final

Table 11: Release Plan for App Repository & Provisioning Support Process Features

4.2.3 App Discovery, Consumption & Re-use Investigation

This process defines necessary features of End-Users, but also App Developers (for re-use), to find and purchase an App within the Store. Basically, this process is comparable to the search in current web shops. An even better analogy is provided by the app stores of big mobile phone operation system provider, as Google or Apple. Based on free text search or categories, the user can find an appropriate App for his or her current demand. The information for specific that is provided by Apple’s or Google’s app store is similar to the information the FIspace Store has to deliver: app descriptions, list of features, user guide, pricing and payment information, rating and user comments as well as additional advertisement material like videos or pictures. In general, we distinguishes two aspects for the App discovery, consumption and Re-use

1. **App Discovery for End-Users**
   We assume that End-Users are primarily interested in finding an App for direct use in order to solve their current business problem. Consequently, they are focusing on the feature description, license and price of the designated App before the purchase. After purchasing an App they probably want to get access to an user manual to understand the App usage.
2. **App Discovery for App Developers**
   In contrast to End-Users, App developers focus on the re-use of existing Apps. For this they need access to feature description of the App and pricing information. However, access to technical information and determine the modalities of a potential reuse are of greater importance compared to End-Users

Under consideration of these different aspects we were able to identify a set of features the App Discovery, Consumption & Re-use Investigation process contributes to the Store. Table 12 summarizes this.

Component	Description	Relevant FIWARE GEs
App Discovery & Re-use Support for Developers	**Tool Support to allow Developers to search, find, and inspect available apps to develop new apps, incl.:** • Search for Apps (UI): repository browsing + advanced search features • Support for detailed investigation: main features, technical interfaces, data models, dependencies, usage terms & conditions, etc. • Ratings of Apps by Consumers & Developers	• Marketplace GE • Discovery & Matchmaking Component • Repository GE • Registry GE

Table 12: Necessary Features for the App Repository & Provisioning Support Process

Based on this feature definition we are able to provide an initial release plan. Table 13 enlists the features together with an overview of the expected milestones.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Overview view of Apps in the Store	UI and User Experience for seeing all available apps, and the overall look & feel of the FIspace Store	Initial	Refined	Final
Detailed view for FIspace App	UI with user features and content for the detailed view of a single FIspace App in the Store (see Figure 19 for initial conceptual design)	Initial	Refined	Final
Search & investigation support for End-Users	Support for finding and investigating FIspace Apps (perspective: End-User)	Initial	Refined	Final
Search & investigation support for App Dev.	Support for finding and investigating FIspace Apps (perspective: App Developer)	Initial	Refined	Final

Table 13: Release Plan for App Discovery, Consumption & Re-use Investigation

4.2.4 Purchase & Revenue Management Support

After an End-User or App developer found a suitable App for his/her demand, he or she needs to purchase it in order to be able to use it. For this the FIspace Store provides the necessary functionality to accept the terms and conditions of an App (e.g. its license), present the price for the purchase and executes the payment and adds the purchased App to the accessible Apps of the user. Consequently, the following aspects for the Purchase & Revenue Management Support can be identified:

1. **Acceptance of Terms & Conditions (by End-User and App developer)**
   After the End-User or the Developer has picked an App, he/she has to choose the appropriate usage terms & conditions. We assume that for the most Apps will be a difference between plain usages of an App by End-Users or if the App shall be re-used by developers. Especially developers have to ensure that right license models etc. are selected. The FIspace Store will present to different terms & conditions and let the user choose. The calculation of the price depends on the selected model.
2. **Payment Conduction**
   After the App developer or End-User has picked a certain set of terms & conditions the payment has to be conducted. The FIspace Store provides the necessary features for this in order to implement the payment via multiple ways.
3. **After Purchase**
   After an App was purchased the FIspace Store has to add the App to the profile of the buying user (developer or End-User). With this other components can determine which Apps are accessible by which user. This is of special importance for the T220 – Front-End, which has to integrate Apps in the user interface.

Under consideration of these aspects, we were able to identify a set of features the Purchase & Revenue Management Support process contributes to the Store. Table 14 summarizes this.

Component	Description	Relevant FIWARE GEs
Purchase of Apps (for Consumers)	**Tool Support for allowing Consumers to ‘buy’ apps, incl.:** • Procedure for purchase process: select items for purchasing, selected & agree on purchase terms & conditions, confirm purchase (‘shopping chart’ style or the like) • Invoicing & payment (via external services?) • Purchase of updates and service packages for apps • Approval of required rights	• Linked USDL • RSS (Revenue Sharing) GE
Financial & Revenue Sharing Mgt	**Manage financials of the FIspace: income by end-user payments, revenue sharing among providers,** incl.: • Tracking of Service & App Usage by Consumers • Selection & Configuration of Pricing & Payment Models • Revenue Sharing among FIspace Operators & Service / Solution Providers • Statistical analysis for Consumers, Operators, Providers	• RSS (Revenue Sharing) GE

Table 14: Necessary Features for the Purchase & Revenue Management Support Process

Based on this feature definition we are able to provide an initial release plan Table 15 enlists the features together with an overview of the expected milestones.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Purchase Procedure & Tool Support	Tool-supported procedure for buying a FIspace App	Initial	Refined	Final
Payment Models	Set of pre-defined payment models for apps, optionally extended with configuration options	-	Initial	Final
Revenue sharing	Support for defining & managing the revenue sharing among FIspace providers	-	Initial	Final
Apps Consumption and Usage Statistics	Various statistics for FIspace users, app providers, and platform operators on usage and sales of FIspace Apps	Initial	Refined	Final

Table 15: Release Plan for Purchase & Revenue Management Support Process Features

4.2.5 Planned Generic Enablers

In the previous section we enlisted the different processes and features of the FIspace Store and already gave an overview which Generic Enablers could be used for which features. In this section we summarize all mentioned Generic Enablers and give some further explanation for them.

4.2.5.1 Linked USDL GE

Linked USDL is a service description language which not only describes the technical aspects of a service but also the business and operational aspects. The language is based on the Resource Description Framework (RDF) and consists of several modules which are mainly optional for a service description. As it is based on RDF, it can be easily extended with additional domain-specific extensions. This means that Linked USDL – as the comprehensive description model for both technical and business services that underlies the Generic Enablers offered for the Application and Service Delivery Framework – can easily be extended for the description of FIspace Apps.

4.2.5.2 Marketplace GE

The Marketplace provides the basic functionalities that are necessary for bringing together offering and demand for making business. This include basics services for registering business entities, publishing and retrieving offerings and demands, search and discover offerings according to specific consumer requirements as well as lateral functions like review, rating and recommendation. However, the Marketplace GE provides only hub functionalities: this means that it is not an (App) store itself, but allows connecting multiple (App) stores in order to provide a single point of access to them. In addition, the Marketplace offers only APIs but no user interface. However, basic functionalities (e.g. shopping basket) needed for implementation of an (App) store are available, although a store implementation itself is not provided so far. Part of the Marketplace GE is also the Discovery & Matchmaking Component which enables the comparison of several service offers and service demands. For an App store mainly the comparison component of it is relevant. So far there is even a user interface available for comparison of service offers (at least for cloud services).

4.2.5.3 Repository GE

The Repository GE should provide a consistent and uniform API to Linked USDL service descriptions and associated media files for applications of the business framework. Service providers can use the Repository GE to publish their Linked USDL descriptions or any other files. However, unfortunately there is no searching functionality available in the repository at all which means that the repository must be connected to a Marketplace GE in order to provide searching. That is the major drawback with the Repository GE at the moment.

4.2.5.4 Registry GE

This provides basic registry facilities that can be used to register offers and demands stored indifferent places like e.g. (external) stores places that connected to the Marketplace GE, in other Linked USDL

repositories, or somewhere else on the Web. The Repository GE is closely integrated with the Marketplace GE.

4.2.5.5 Store from FIWARE Discovery & Matchmaking Component

The development of the Store GE was frozen in the FI-WARE project for a long time which means, that – compared to the other GEs – the development progress of the Store GE way behind the progress of the other GEs. Thus, there are not many technical details available about this GE so far. However, it is expected from the FIspace perspective, that it can be reused or at least serve as inspiration for implementation of the FIspace store.

4.2.5.6 Store Integration of WireCloud GE

The WireCloud GE is not foreseen to be used directly for integrating the FIspace Store, but rather could be used to for the Front-End integration. For this reason we want to mention it here. As described by the WireCloud development team, the GE provides a store component for trading portal widgets. For this, the WireCloud store builds upon the Repository and the Marketplace GE. The FIspace store also evaluates those GEs as possible candidates to facilitate the implementation and we currently investigate how and to which extend the FIspace Store can rely on the WireCloud Store. We currently assume an easy integration due the use of the same backend for both stores. However, the Front-End team has not finally decided about the usage of WireCloud. We are currently in tight cooperation in order to further evaluate WireCloud.

4.3 The B2B Collaboration Core Modules

The following provides the initial technical design of the two modules – namely the Business Collaboration Module (BCM) and the Event Processing Module (EPM) – that constitute the basis for enabling seamless and coordinated interaction among businesses working in a cross-organizational collaborative business processes in an event-driven manner via the FIspace and FIspace Apps. As in the preceding sections, we define the main features along with an initial development and release plan and a conceptual architecture along with the technology choice including planned GE usage.

4.3.1 Overview & Main Features

The aim of this module is to create, manage, execute, and monitor collaborative processes in the FIspace platform. To this end, two complementary components will be implemented: The Business Collaboration Module (BCM) and the Event Processing Module (EPM).

The BCM component is the responsible to orchestrate the different processes from different stakeholders and assure the correct sequence of the tasks execution. The BCM is based on the entity-centric approach [6]. This approach relies on the notion of *entities* (aka, as business entities, artifacts, or dynamic artifacts, or business collaboration objects). These provide a holistic marriage of data and process, both treated as first-class citizens, as the basic building block for modeling, specifying, and implementing services and business processes. A (business) entity is a key conceptual concept that evolves as it moves through a business (or other) process. An entity type includes both a data schema and a lifecycle schema which are tightly linked. The data schema provides an end-to-end conceptual view of the key data for this entity type. The lifecycle schema of an entity type specifies the different ways that an entity instance might evolve as it moves through the overall process. In FIspace we will use the GSM (Guards, Stages, and Milestones) model to specify the lifecycle schema of the business entities [6].

The Event Processing Module (EPM) component monitors events and detect situations of interest, i.e. situations that require appropriate reactions. The events sources (aka events producers) can be the actual execution of the collaboration (i.e. the BCM), external systems, or sensors. The EPM processes these events and by applying pattern matching derives situations of interest (for a background on event processing refer to [7]). Examples of situations of interest can be: Missing documentation at a certain point in time, a sensor reading outside a permitted range, a delay in a delivery. In general, we can distinct between situations that result from the actual execution of the process or collaboration and situations that result from external events (i.e. events coming from external systems or sensors).

The EPM in FIspace supports two types of situation detection capabilities: reactive and proactive. Reactive rules analyze past events and derive situations by applying pattern matching over a single or a set of events over time. Proactive rules, on the other hand, relate to situations that are likely to happen in the (near) future. In general, we refer to proactive event-driven computing as the ability to mitigate or eliminate undesired states, or capitalize on predicted opportunities—in advance. This is accomplished through the online forecasting of future events, the analysis of events coming from many sources, and the application of online decision-making processes. For background on proactive event-driven computing refer e.g. [8], [9]; for reports on application studies see e.g. [10], [11].

4.3.2 High-level Technical Architecture

As mentioned above, the B2B Core Modules are based on two major technical building blocks or components: The BCM and the EPM. Both modules communicate via a Publish/Subscribe Component. This component is an Enterprise Service Bus (ESB) to be provided by T260 “Operating Environment”, specifically ST263 “Enterprise Service Bus Development”. External systems can be any kind of backend systems or sensors connected to FIspace, assumable by the System and Data Integration Module.

Figure 20 shows the high level technical architecture in a UML component diagram[11]. In the following subsections, we elaborate on the sub components and their interplay.

[11] UML Component Diagram Syntax briefly: Rectangles represent components. Contained rectangles represent sub-components. Offered interfaces are visualized with lollipops. Consumed interfaces are indicated with semi-circles.

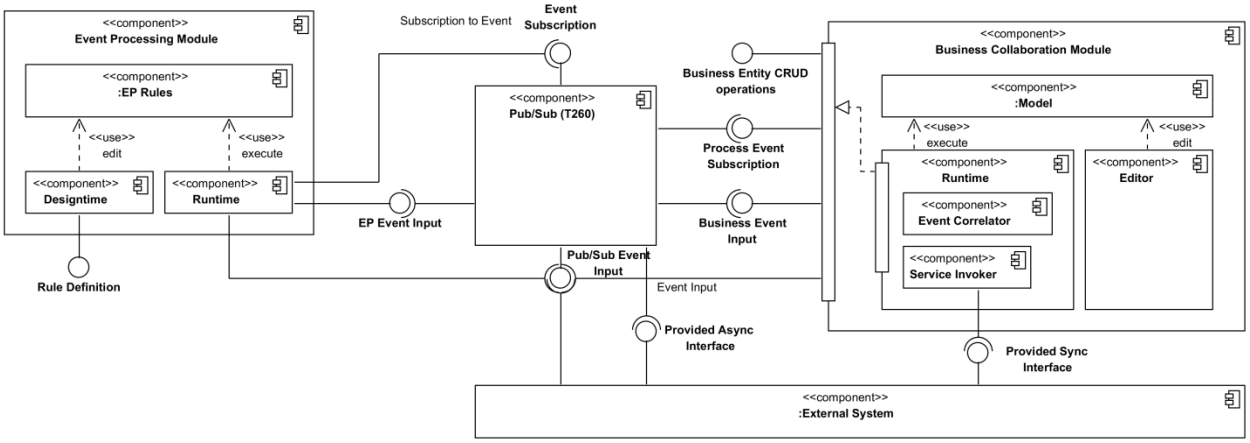


Figure 20: B2B Core Modules high level technical architecture

4.3.3 B2B Collaboration Support

The Business Collaboration Module (BCM) takes care about the execution of collaborative business processes among FIspace users. To this end, it offers a design-time and a run-time component shown as the *Editor* and *Runtime* components respectively in Figure 20 above. The design-time component is used for defining process models (i.e. data and lifecycle schemas), and the runtime component is used for instantiating and executing the defined process models.

4.3.3.1 BCM Design-time Component

The design-time component of the BCM supports the definition of the data and lifecycle schemas of business entities. These definitions are stored in the *Model* component as shown in Figure 20.

The data schema includes all the information relevant to the business entity type. In general the data schema (aka information model) holds three types of information: attributes related to the entity type (business attributes), attributes related to the state of the entity (statuses of milestones and guards), and services/events attributes.

The lifecycle encapsulates the behavior of a Business Entity (BE). The lifecycle schema applied in FIspace is based on the GSM model. As the name implies, GSM is defined around the notion of Guards, Stages, and Milestones. Stages represent clusters of activity and are visualized as rectangles with rounded corners. A Stage can either contain further sub-Stages, representing sub-activities, or it can be atomic, which means that it contains a task that is executed upon activation. Each Stage can own one or more Guards, which control its activation. Guards are defined as conditions of the form “on <event> if <condition>”, where event is a placeholder for the occurrence of an arbitrary process relevant event, and condition a place holder for an arbitrary condition expression against the state of the described BE. The visual element for Guards is a diamond at the left side of their owning stage. Each Stage can have one or more Milestones. They represent the achievement of distinct business objectives and are visually represented as circles at the right side of their owning Stage. Like Guards, they are noted in the form “on <event> if <condition>”. The achievement of a Milestone closes a Stage and is often used as part of a Guard-condition in another Stage of the BE lifecycle. Figure 21 illustrates the constructs and graphical elements of GSM lifecycles.

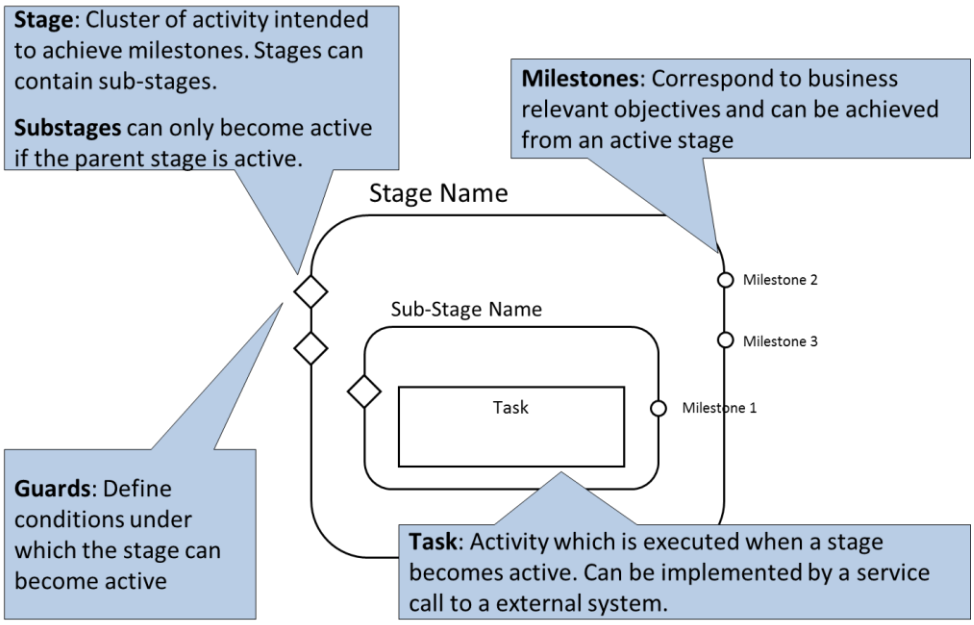


Figure 21: GSM constructs

A BE evolves through its lifecycle by events that happen or state modifications. These can be originated by the external world or by another Stage. In GSM, this is transformed into events or services that activate/deactivate stages. The event-driven nature of the GSM model implies for a natural conceptual interplay with the EPM. The GSM model controls the flow of the events and eventually the flow of the collaborative process. The BCM provides design-time support via a graphical user interface illustrated in Figure 22 below.

4.3.3.2 BCM Run-time Component

The run-time component of the BCM supports the instantiation and execution of collaborative processes. It instantiates and executes BE types, captured through the design-time component.

BEs are the main building block of collaborative processes in the FIspace platform. BE data and state attributes need to be accessible for external applications, which require awareness of the involved BEs states. Accordingly, a “Create Read Update Delete” (CRUD) API will provide services to interact with BE instances (*Business Entity CRUD operations*).

BE lifecycles can make use of events in the conditions of the guards and milestones. In order to receive the relevant events, the BCM has to have a subscription on the relevant event types in the central ESB component, which will distribute messages based on the “Publish/Subscribe” paradigm. In order to receive the incoming events, the BCM needs to offer a corresponding *Business Event Input* interface.

An open design question is, whether the BCM will be responsible for creating the event subscriptions in the ESB, or if a central manager component will take care about this.

The BCM will send 1-way messages, and make asynchronous 2-way service calls through the central ESB. For this it will consume the ESB’s message input interface (*Pub/Sub Event Input*).

While asynchronous service calls will be done through the ESB infrastructure, synchronous service invocations will be executed directly between target system and the BCM by the *Service Invoker* component through the *Provided Sync Interface*. The corresponding service definitions and endpoints are configured during design time.

Figure 20 above shows two sub-components of the runtime, which are considered as noteworthy in this context. More components will be described after elaborating further on the architecture.

The Event Correlator is responsible for asynchronous communication with the outside world. It sends out messages of 2-way service calls and takes care of correlating incoming messages back to their initial invocation context. Upon the arrival of a response it further decides whether:

- An update of BE attribute values has to be done based on the supplied data in the response
- A new BE instance has to be instantiated and initialized
- A business event has to be inserted into the BE container. If yes, it determines the event type and payload

4.3.3.3 Technology Choice

For the BCM engine, the prototype of the former ACSI EU-project (http://www.acsi-project.eu/) will be used. It provides a design and runtime environment for BEs with GSM lifecycles and is available as open source under the Apache Software Foundation License v2.0. ACSI is a Java based web-application and can be hosted in Java Servlet containers. Figure 22 shows a screenshot of the ACSI web user interface with an open instance of the BE lifecycle editor.

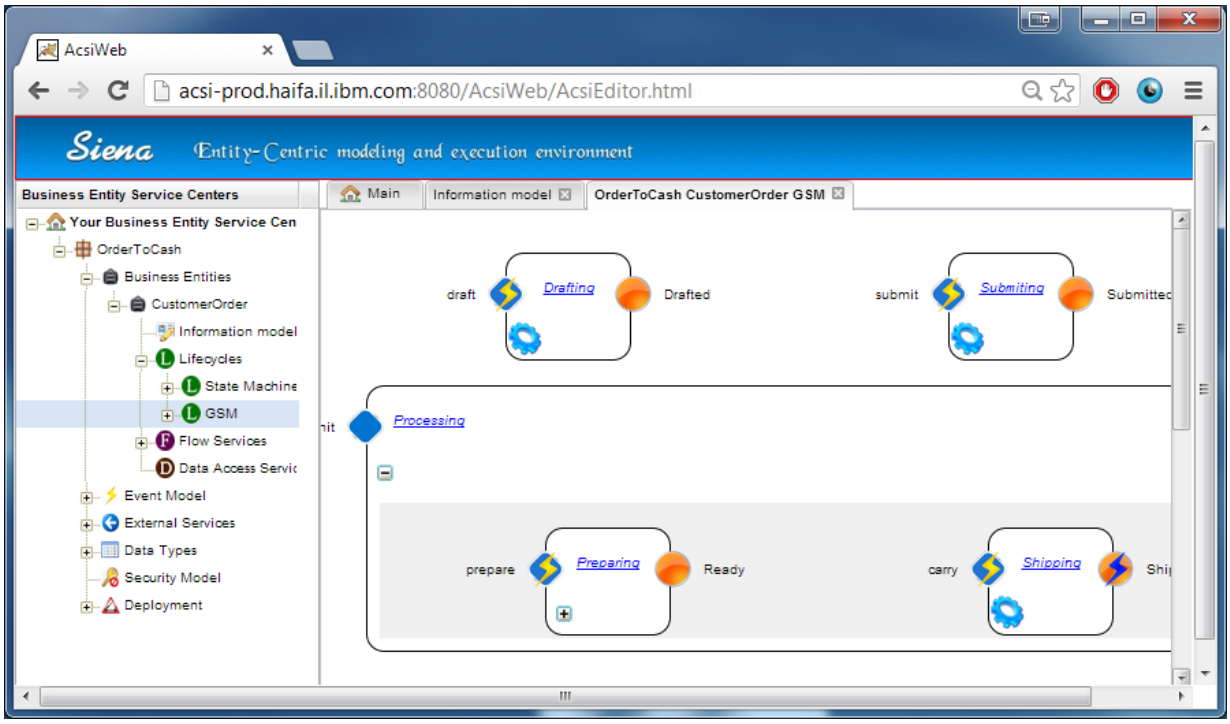


Figure 22: ACSI Business Entity Lifecycle Editor

4.3.3.4 Release plan

In this section we provide an initial plan of activities around the BCM component. It is elaborated based upon our current understanding of the required activities and our current effort estimation. The table below provides an overview of the envisioned activities and their due date.

Table 16: Initial Release Plan Business Collaboration Module

Activity	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
ACSI Development Environment & Deployment Environment	1) Setup of a development environment with the ACSI BCM prototype sources. 2) Setup of a test deployment environment for the ACSI BCM instance, which will be used for development.	Final	-	-

Technical Investigation of ACSI’s architecture and interfaces	FIspace will use interfaces of the ACSI BCM. Furthermore, ACSI will probably extend or change some aspects of the BCM. Therefore, technical knowledge has to be built up.	Mature	Advanced	
Process Modeling	This activity consists out of two sub-activities: 1) Syncs with the trial partners will be held. Subjects are the processes, which should be supported by the BCM. 2) Formal modeling of the discovered processes in the GSM language	First Draft	All Trials Stable	All Trials Final
Flexible processes - Requirements	FIspace plans to develop a component for flexible process composition and (runtime) adaption. In order to gain a better understanding of real end user requirements with regard to the mentioned points, an elicitation of requirements will be conducted.	Stable	Final	-
Flexible processes – Implementation	Implementation of the component for enabling users to customize processes according the previously task requirements	-	Initial	Final

4.3.4 Event Processing Component

The terminology used in this report and in T240 is based on, and follows, the event processing language as presented in [7]. For proactive event-driven computing refer to e.g. [9], [11], [12].

Generally speaking, 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. The word “event” is also used to mean a programming entity that represents such an occurrence in a computing system. In the latter definition, an event is an object of an event type. Events are actual instances of the event types and have specific values. For example, the event "today at 10 PM a customer named John Doe booked a new order" is an instance of the Order event type.

Entities connected to the EP engine can play two different roles: the role of *event producers* or the role of *event consumers*.

An event producer is an entity at the edge of an event processing system that introduces events into the system. Event producers are the source of events for event processing. An event consumer is an entity at the edge of an event processing system that receives events from the system. Event consumers are the sink point of events.

In the FIsapce, we identify three event producers,

1. *BCM (Business Collaboration Module):* events related to the actual execution of a collaboration.
2. *IoT (Internet of Things):* events received from sensors like RFID tags or GPS systems.
3. *Backend systems:* events originated from systems external to FIspace, such as booking systems.

and two consumers of FIspace EPM events,

1. *BCM:* events related to actual execution plans, such as pickup arrival, meaningful deviation in the quality of a plant, and schedule delay. The BCM, in turn, might transmit some of these events as notifications to the user via FIspace frontend.
2. *FIspace frontend:* proactive notifications regarding future probable events not directly related to the actual/real-time execution of the transport plan.

As aforementioned, the architecture envisioned for the B2B Core Modules includes an ESB component. This bus will serve as the channel via events will be pushed by the EP engine to reach out potential consumers and from which events will enter the EP engine from producers. This ESB will use a Pub-

lish/Subscribe mechanism. In this interaction pattern, a consumer registers to specific events (subscriptions) and a producer pushes events to the bus (publish) via the *Pub/Sub Event Input API*. The Pub/Sub component will be provide by T260.

The EPM includes the following two conceptual components:

- A design-time tool (aka authoring tool) to define event processing applications on data interpreted as events (incoming events from producers)
- A run-time engine that processes events as they occur and generate derived events accordingly. When these derived events are detected situations, these are emitted to the events consumers defined in the system.

The design and run-time tools are further detailed in the following sub-sections.

4.3.4.1 EPM Design-time component

The event processing definitions file, i.e. the event types and rules to be implemented, can be created in three ways:

1. Build-time user interface – By this, the application developer creates the building blocks of the application definitions. This is done by filling up forms without the need to write any code. The file that is generated is exported in a JSON (JavaScript Object Notation) format to the CEP run-time engine.

2. Programming – The JSON definitions file can alternatively be generated programmatically by an external application and fed into the CEP run-time engine.

3. Manually – The JSON file is created manually and fed into the CEP run-time engine.

The EPM authoring tool realizes option 1 via the *Designtime* component and the *Rule Definition API*. The application developer defines all the definitions using a form based user interface to define the event types and rules. A screenshot of the form based UI is shown in Figure 23 below.

The JSON file that is created at build-time contains all Event Processing Network (EPN) definitions, including definitions for event types, Event Processing Agents (EPAs), Proactive Agenst (PRAs), contexts, producers, and consumers. Once the application definitions are identified and defined, there is a need to subscribe to the relevant expected input event types (via the *Subscription to Event API*). Again, as in the case of the BCM, the issue of the subscriptions definitions to the ESB is still open to be addressed in the scope of T260.

4.3.4.2 EPM Runtime component

The EP runtime engine is the heart of the module – it receives events from producers, monitors and checks them for predefined patterns, and produces derived events. When the latter are emitted outside the module to consumers, they are called detected situations.

The application definitions, i.e. the EPN including the definitions of the producers and consumers, are written by the application developer during the build-time using the authoring tool. The definitions output, in JSON format, is stored in the *EP Rules* component and then sent to the EP run-time engine during execution.

The EP *Runtime* has three main interfaces: one for getting input events using input adapters, a second for sending output events using output adapters, and a third for getting application specific definitions. In the case of a RESTful API for the first interface, the EP engine exposes RESTful services for to enable external applications to send events into the event processing engine. In the case of RESTful API for the output interface, the EP engine can use external applications APIs to consume the events emitted from the EP engine.

During execution time, the consumers and producers definitions are translated into input and output adapters. The physical entities representing the logical entities of producers and consumers in EP are adapter instances. For each producer, an input adapter is defined, which defines how to send the data from the source resource and how to format the data into EP's object format before delivering it to the

run-time engine. The adapter is environment-agnostic, but uses the environment-specific connector object, injected into the adapter during its creation, to connect to EP runtime.

At execution, the run-time accesses the metadata file, loads and parses all the definitions, creates a thread per each input and output adapter, and waits for events incoming from the input adapters (producers) and forwards events to output adapters (consumers). The incoming events are received via the ESB by the subscribe mechanism thorugh the input adapters to be processed by the run-time engine.

The consumers and their respective output adapters operate in a push mode – each time an event is published by the runtime it is pushed through environment-specific server connectors to the appropriate consumers, represented by their output adapters, which publish the event in the appropriate format to the designated resource via the ESB.

4.3.4.3 Generic Enablers

The EPM module will build upon FI-WARE Proton (IBM Proactive Technology Online) CEP GE[12] . The run-time engine supports a RESTful, resource-oriented API accessed via HTTP that uses either XML-based, JSON-based, or tag-delimited format representations for getting input events and for sending derived events.

The EPM will fully exploit the CEP GE and extends it to cope with proactive capabilities by adding building blocks called PRA (PRoactive agents) to the EPN. The CEP GE has been already successfully tested and applied during the phase 1 of the FIspace project. In phase 2 we plan to implement all reactive rules for FIspace apps exploiting the functionalities provided by the CEP GE. For the sake of proactive event-driven applications, the CEP GE will be extended and additional building blocks will be added to it. For the proposed architecture of the EPM including proactive extensions refer to Deliverable 6.5 in FIspace project “Final technical specification and phase 2 implementation plan for the event processing component”[13].

4.3.5 Technology Choice

In Task 240 two engines have been chosen to implement the CEP and EP as detailed in the following sections.

As mentioned in the GE section, the EPM will be built on the CEP GE. Figure 23 shows a screenshot of the CEP GE build-time tool.

[12] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.CEP and

[13] Available at http://www.finest-ppp.eu/

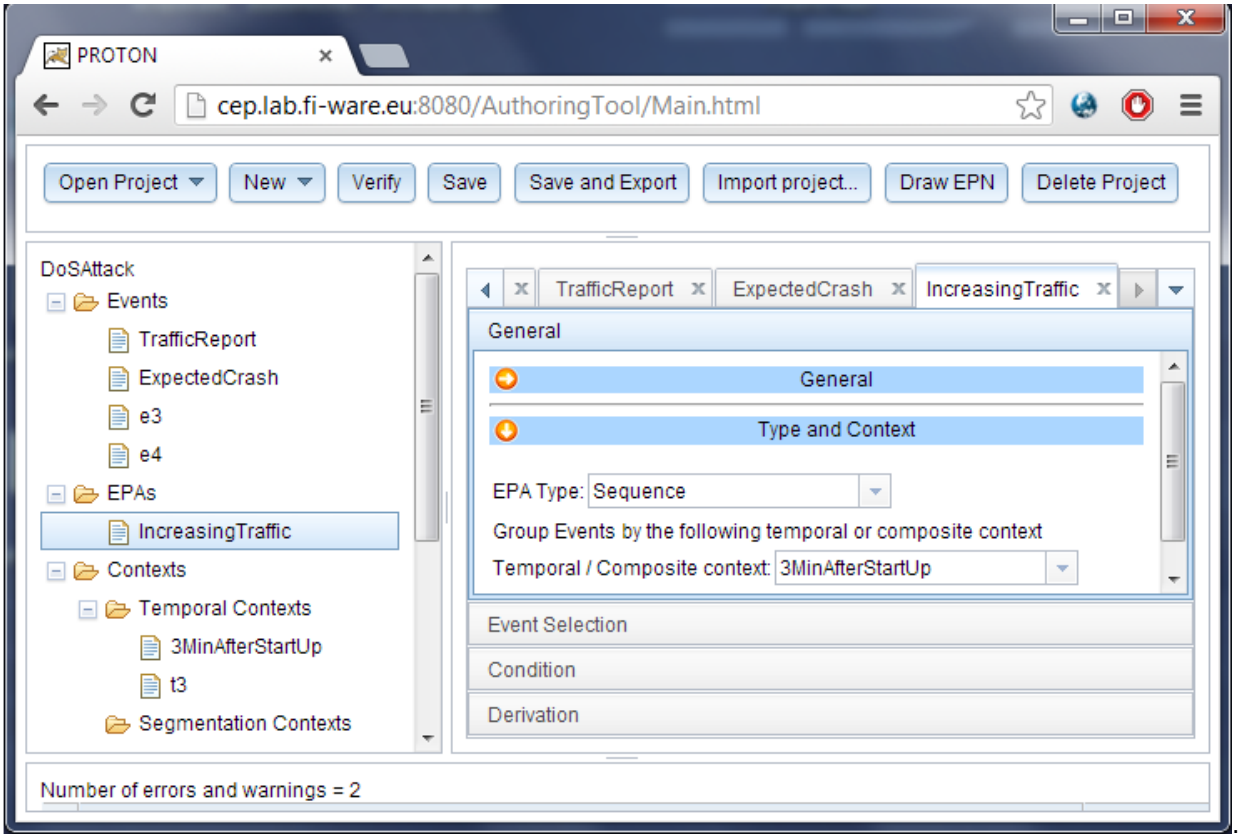


Figure 23: Screenshot of Proton CEP GE authoring tool

4.3.5.1 Release Plan

In this section we provide an initial plan of activities around the EPM component. It is elaborated based upon our current understanding of the required activities and our current effort estimation. The following table provides an overview of the envisioned activities and their due date.

Table 17: Initial Release Plan Event Processing Module

Activity	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
CEP GE Development Environment & Deployment Environment	1) Setup of a development environment 2) Setup of a test deployment environment for the CEP GE EPM instance, which will be used for development.	Final	-	-
Technical Investigation of CEP GE architecture and interfaces	The EPM will use CEP GE as the underlying design-time and run-time tools; therefore a deep understanding of the available code is required.	Mature	Advanced	
Rules definitions	This activity consists out of two sub-activities: 1) Syncs with the trial partners will be held. Subjects are the event rules, which should be supported by the EPM. 2) Formal specification of the event rules applicable to the trials	All First Draft	All Trials Stable	All Trials Final

Flexible rules definition - Requirements	FIspace plans to develop a component for flexible rules definition to ease the creation of new rules using the EPM. A requirements analysis and deployment plan will be established based on the project trials characteristics.	Stable	Final	-
Flexible rules definition – Implementation	Implementation of the flexible events definition component according the previously task requirements	-	Initial	Final
Proactive component - Requirement	The CEP GE will be extended to deal with proactive event-driven applications. A requirement analysis will be conducted based on the project trials and candidate scenarios will be chosen.	Stable	Final	-
Proactive component - implementation	Implementation of the candidate scenarios chosen for proactivity testing.	-	Initial	Final

4.4 System & Data Integration

This section aims at providing a general overview of how system and data integration will be achieved within the FIspace project. We begin by introducing core functionalities along with potential Generic Enablers and technologies that could support the system and data integration process. In the final section, we present the conceptual architecture.

The overarching purpose of T250: System and Data Integration is to provide a robust and scalable infrastructure that enables seamless integration of external legacy systems/IoT systems with the FIspace platform and applications deployed on it. Outputs from the task will facilitate the implementation of Web based, FIspace-driven applications by providing unifying data models, data mediation tools and system integration APIs.

Some of the key objectives of this task are:

- Define the technical specification (UML artifacts) for system and data integration in FIspace.
- Propose system architectures at varying levels of granularity to address the various aspects of data mediation and system integration.
- Identify and implement data exchange standards and communication protocols for ensuring interoperability between external systems/IoT systems integrated via the FIspace platform.
- Analyze legacy system interface specifications with the aim of defining system integration APIs.
- Identify functional and non-functional requirements of potential applications to be implemented and integrated via FIspace.
- Exploit existing technical system specifications available via the trial systems in the project for development of the interfaces as part of the integration infrastructure.
- Define unifying data models and mapping heuristics that enable data integration with focus on semantic rather than syntactic interoperability.
- Identify/implement data mediation tools that facilitate alignments between legacy data models.
- Provide RESTful Web services that enable system integration.
- Investigate the currently available Generic Enablers from the Fi-ware project, in order to identify which of them are functional and suitable to be used for System and Data integration purposes. This will extend the functionality provided by the tools to be built, enable technology reuse, and allow more external systems to be integrated.

4.4.1 Business and Legacy System Integration

This section describes the main features and first results of Sub-Task ST252 – Business and Legacy System Integration. It comprises a general overview and description of the main features elaborated so far, an initial assessment of potential Generic Enablers as well as other technologies for the implementation, and a preliminary release plan. We address each of those aspects in a separate subsection below.

4.4.1.1 Overview & Main Features

Subtask ST252 is concerned with the definition and implementation of communication channels between the FIspace and external business and legacy systems (e.g. in-house logistics solutions, ERP systems, farm management systems). In particular, this shall include:

- Defining a set of standard channels
- Tool-supported mechanisms to allow the easy creation of adapters to business and legacy systems
- APIs for importing / exporting data from connected business & legacy systems into the FIspace

Different potential integration scenarios that are currently planned to be supported by the FIspace infrastructure for Business and Legacy System Integration are indicated in Figure 24and described in the following.

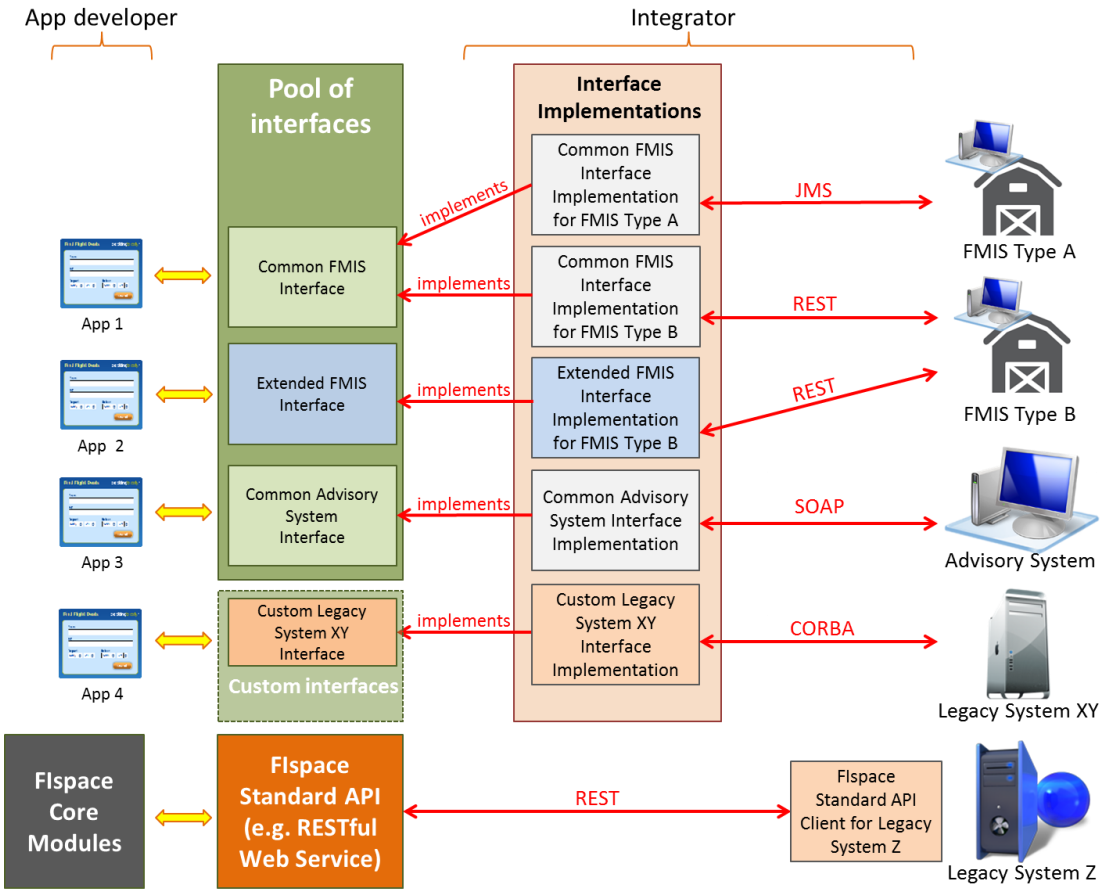


Figure 24: Potential Business and Legacy System Integration Scenarios

The FIspace System and Data Integration module plans to provide App developers with a set of abstract interface definitions that will provide APIs to access the most commonly used legacy systems. Each interface will bundle common functionality for one "class" of legacy system, such as Farm Management Information Systems (FMIS) or Advisory Systems: see "Common FMIS Interface" and "Common Advisory System Interface" in Figure 24.

Each interface in turn is supposed to have one or more concrete implementation(s) that handle the actual communication and data exchange with a concrete instance of the legacy system class. For example, as depicted in Figure 24, there could be two types of FMIS (Type A and B) to be accessed by App 1, each one allowing the same kind of data access but through a different kind of interface and communication protocol, e.g. JMS and REST in Figure 24. Consequently, there should be two concrete implementations: "Common FMIS Interface Implementation for FMIS type A" and "Common FMIS Interface Implementation for FMIS type B". Such concrete implementations will have to be developed by system integrators for each specific type of external system, and this implementation work is planned to be supported by tools to be developed in the scope of T250. The development of a few reference implementations of interfaces (to access some of the legacy systems used in the FIspace trials) is also planned.

In case an app needs specific functionality to access an FMIS, and that functionality is not yet covered by the "Common FMIS Interface", it should be possible to extend the "Common FMIS Interface" with additional/custom methods. This is indicated by the communication between App 2 and FMIS Type B in Figure 24. Consequently, an extended interface implementation is needed, which is indicated by the "Extended FMIS Interface Implementation for FMIS type B" in Figure 24.

App 3 needs information from an existing advisory system, therefore it is built to access the "Common Advisory System Interface" from the FIspace pool of interfaces. To actually access the existing advisory systems, a specific implementation for each type of advisory system is needed (indicated by "Common Advisory System Interface Implementation".

In case a new app (App 4 in Figure 24) requires access to some legacy system that is not yet covered by the existing interfaces from the pool of interfaces, it should be possible for a developer to create his/her own custom interface and a corresponding implementation (indicated by "Custom Legacy System XY Interface" and "Custom Legacy System XY Interface Implementation" in Figure 24).

The FIspace System and Data Integration module should also support the use case that some legacy system (“Legacy System Z” in Figure 24) would need to “push” data into the FIspace platform or notify a FIspace App (or a FIspace Core Module) about certain events. In order to realize this, the System and Data Integration component would need to provide a “FIspace Standard API” (e.g. as a RESTful Web Service). The legacy system owner would then need to implement the corresponding client on the side of the legacy system (see “FIspace Standard API Client for Legacy System Z” in Figure 24).

4.4.1.2 Potential Generic Enablers

In this section, we present our initial assessment of Generic Enablers as well as other technologies potentially to be used for an implementation of the envisioned features of the Business and Legacy System Integration (Sub-Task ST252).

- **Publish/Subscribe Context Broker**

The Orion Context Broker is an implementation of the Publish/Subscribe Context Broker GE. Using the NGSI9/10 interfaces, clients can do several operations like:

- Register context producer applications
- Update context information
- Being notified when changes on context information take place or with a given frequency
- Query context information.

This kind of generic enabler is useful in data/context scenarios where a component is needed in order to play the role of the broker between applications that produce data, and applications that consume it.

This Generic Enabler could possibly be used by ST252 – Business and Legacy System Integration to allow FIspace core modules and apps to subscribe to specific types of data provided by legacy systems. When new data would then be published (i.e. pushed into FIspace) by existing systems, this could be provided directly to those apps that subscribed to the respective type of information.

4.4.1.3 Further technologies under consideration

Related to the FIspace standard API as well as the tool support for creation of interface implementations to access existing systems, several Java-based tools are currently under consideration that support the creation of Web Services and Web service clients, e.g. Jersey (https://jersey.java.net/) and Apache CXF (http://cxf.apache.org/).

4.4.1.4 Release Plan

In this section, we provide an initial release plan for Sub-Task ST252 – Business and Legacy System Integration. It is elaborated based upon our current understanding of the required features and our current effort estimation.

Table 18: Release Plan Business & Legacy System Integration

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Pool of Common Interfaces	Management of a pool of standard interfaces to be accessed by FIspace apps	Initial	Refined	Final
Discovery of Interface Implementations	Option of discovery and binding of interface implementations		Initial	Final
FIspace Standard API	FIspace standard API for external systems to push data into FIspace	Initial	Refined	Final
Subscription mechanism	Subscription mechanism for FIspace components to get data provided by external systems	Initial	Refined	Final
Tool support	Tool support to allow the creation of standard interface implementations accessing specific business and legacy systems		Initial	Final

4.4.2 Data Handling and integration

This task is concerned with the management and integration of incoming/ outgoing data during communication between the FISpace platform and an external system/sensor offering interoperability among the integration process. System integration, data mediation and transformation are facilitated by deploying a well-researched set of generic enablers, standards, third party components and a scalable system architecture.

4.4.2.1 Overview and Main features

The key objectives of Task 253 include:

- Define and implement a common data model for data exchange between the FISpace Apps connected data sources, legacy systems and services.
- Define and implement adapters, in charge to offer bridging from the different data sources to the FISpace platform facilitating heterogeneous data sources/systems integration having into account di-verse technologies like ontologies.
- Define and implement transformers, in charge to handle the data transformations between message formats.

FIspace enables the integration of Web applications developed using the FIspace SDK or otherwise with external legacy/IoT systems, by providing a set of supporting abstract interfaces. The requirements for defining the interfaces are derived by analysing a repository of existing system specifications which include interface definitions of the systems deployed by the trials participating in the project.

Messages received from external applications/sensors may need to be transformed into appropriate data formats that can be consumed by the external systems and vice versa. The transformation process incorporates data mediation tools for mapping between the data formats prescribed by FIspace and those supported by the external systems.

As an example, consider the development of a Web based application for the farming domain. The application requests advise on crops from agricultural expert systems, deployed externally to FIspace. The integration between the Web application and the external expert system is facilitated via message exchange through FIspace. A set of interfaces, corresponding to various external systems, are provided to the application developers.

A typical interface for an expert system would include method signatures such as:

selectProductForAdvice();

provideDetailsOfRequest(String details);

provideSensorData(SensorDataStructure sensorData)

getCostOfAdvice();

Implementation of the interfaces, as RESTful Web services are deployed on FIspace. The Web services accept messages from the requesting Web applications and routes them to the designated external systems.

4.4.2.2 Potential GenericEnablers

Mediator GE

The Mediator is a middleware application responsible for providing interoperability among different communication protocols and among different data models.

Some of the functionalities offered include

- Exposing a REST web service as a SOAP web service
- Exposing a service with an xml payload with any different xml structure for the payload
- Exposing old ASCII delimited message used through old protocols such as FTP, as web services with an xml payload, both SOAP or REST

This Generic Enabler is based on WSO2 Carbon and Apache Camel project. WSO2 Carbon is a middleware platform, on top of which various components are built, including a lightweight Enterprise Service

Bus. It provides many useful features for integration purposes such as message and service mediation, transforming different data formats or incompatible communication protocols. The mediation functionality can be extended by writing custom mediator tasks.
Its current release relies heavily on the above-mentioned products, e.g configuration of the WSO2 Carbon is made through its management console, there is no Mediator configuration API yet.

The Mediator GE has been successfully tested and it seems to be working.

It can facilitate the communication of the FIspace apps with the external systems by providing data transformation and proxy services. For example, the Mediator can be utilized to allow an App to communicate to any external system only in a RESTful way, leading to the simplification of the development and instantiation process.

4.4.2.3 Release Plan

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Common data model	Define and implement a common data model for data exchange between the FIspace Apps connected data sources, legacy systems and services involved	initial	refined	final
Data Adapters	Define and implement adapters, in charge to offer bridging from the different data sources to the FIspace platform facilitating heterogeneous data sources/systems integration having into account diverse technologies like ontologies.		initial	final
Data transformers	Define and implement transformers, in charge to handle the data transformations between message formats.		Initial	final

Table 19: Release Plan Data Handling and Integration

4.4.3 IoT System Integration

This section aims at providing a description on how the Internet of Things integration could be realized, by giving a general overview and identifying main features that should be supported. We then proceed to provide an initial assessment of potential Generic Enablers and finally a preliminary release plan for the IoT integration.

4.4.3.1 Overview& Main Features

Subtask 254 deals with Internet of Things System Integration, which shall provide the mechanisms and tools so that FIspace applications can have access to external IoT systems and consume data both on demand as well as being able to be subscribed to events coming from the outside world.

Internet of Things builds upon the so called smart objects, which basically are physical things with a small electronic device attached, allowing local intelligence and connectivity to the internet, bridging the gap between the physical and digital world. The vast numbers of these devices expected to be deployed and their interconnection through low-power means will leverage applications that will greatly benefit almost every aspect of human activity, a vision that is slowly but steadily becoming a reality.

Integrating IoT systems is no easy task however, mainly due to the heterogeneity of the enabling technologies, protocols, and data models. For this reason, we will start the integration process by first seeking to provide support for protocols and data models that are used by the various trials, and then move on to incorporate others.

Below, we present a non-exhaustive list and a short description of protocols/standards that will be supported.

EPCIS

Almost exclusively used in conjunction with RFID, the Electronic Product Code (EPC) is a standard from EPCglobal widely used today to uniquely indentify physical objects. It is the basis for the information flow

in an EPCglobal Network, a network consisting of several components, in which data about RFID tagged products is shared, between the interested parties. EPC Information Services (EPCIS) is the standard enabling this EPC-related information sharing, by defining standard interfaces for the capturing and querying of data in EPCIS databases, along with an associated data model. Our goal here would be to provide incorporate these interfaces to allow a FIspace app to extract EPC information from an IoT system’s EPCIS repository (query control and query callback interfaces) and/or be able to receive EPC events from a tag Reader and push them to an EPCIS repository.

CoAP

Constrained Application Protocol is an application layer protocol targeting resource-constrained devices such as low power sensor nodes, and various other embedded devices which will be first class citizens in the Internet of Things. CoAP’s aim is to introduce the web service paradigm into networks of smart objects, and can be thought of as modified version of the HTTP protocol, including several of its functionalities, but redesigned taking into account the low processing power and energy consumption constrains of small embedded devices. The goal is to expose every sensor node’s sensing capabilities as RESTful resources that can be requested by CoAP or HTTP (with HTTP-CoAP mapping) clients. Although currently a draft being developed by the Constrained RESTful environments (CoRE) working group of the Internet Engineering Task Force (IETF), it is worth being considered and examined, as it is expected to have a big impact in the following years. One downside of being a draft specification is that there are no rock stable, mature implementations yet.
However, we feel that FIspace apps should be able to perform requests on devices exposing their measurement capabilities in a RESTful way. For this reason API definitions tailored to FIspace context are considered to be provided that applications can use to obtain available values.

NGSI

Open Mobile Alliance (OMA) Next Generation Service Interfaces (NGSI) are a group of specifications for interfaces covering various areas of functionality. NGSI 9/10 are the ones dealing with context management. The FI-Ware project has provided a RESTful binding of these two interface definitions using XML, along with some convenience operations not originally included in the specification. The central aspect of the information model used is the so called “entity”, which is a virtual representation of a physical object. Each entity has an id including name and type, attributes which model the state of the entity, and optional metadata of these attributes. The values of the defined attributes are the context information that applications exploit in order to support context-awareness. Information about entities is exchanged using a container called “context element”. NGSI 10 is designed for the exchange of context information itself, while NGSI 9 is targeted to the exchange of information, about the availability of Context information and entities.
NGSI support is necessary in order for the apps to be able to communicate with the deployed Generic Enablers, since NGSI has been selected as the main interface for exchanging information and events, and is widely used within the architecture of Internet of Things Service Enablement chapter of the FI-Ware project.

4.4.3.2 Potential Generic Enablers

In this section we present our initial assessment of relevant Generic Enablers we feel that would help us towards integrating Internet of Things devices and events into FIspace. More Generic Enablers will be investigated and tested as they become available, so the following list is far from being exhaustive.

Gateway Data Handling GE

This Generic Enabler is a fast, versatile Complex Event Processor able to collect vast amounts of asynchronous events of different types and correlate them into single events, called Complex Events. It can read from and write to numerous different channels using various different protocols.

In the Internet of Things, systems will have to deal with an ever-growing amount of data from hundreds and thousands of sensors and devices. Millions of readings of a heterogeneous nature, such as temperature, status or any type of readings have to be processed to meaningful information.

The Gateway Data Handling GE is also the first stage of intelligence transforming data into events using smart rules. Applications are now able to collect in real-time large amounts of data, but only relevant data avoiding boring and asynchronous data analysis. It does not only manage raw data, it also allows the definition of some local rules to add value on raw data and send only relevant events when a typical situation happens.

Although three versions were originally released (mobile, OSGI bundle, and servlet), only the servlet version will be maintained and updated. Being based on the Esper java library, it consists of several components, but its core functionality is the Complex Event Processor, handling real time events. Other components include Local Storage, which is needed for the asynchronous communication with “sleeping” IoT devices and pub/sub component, with which it can communicate with other GEs using NGSI, or handle subscriptions of events. One can define rules for grouping, aggregating, sorting filtering of events, using an SQL-like language called EPL.

Among other usage scenarios, it can be utilized by apps in order to transform individual events coming from external systems into knowledge, with each app defining its own EPL rules.

There are many possible uses of this generic enabler depending on various usage scenarios, for example in an advice FIspace app, the publish/subscribe broker component could be utilized in order for the expert system to get the needed measurement values from an farm management system. In addition, the external expert system could define its own EPL rules and event types, so that it can be notified about the conditions in the farm in a high level way. An example of this would be that the raw sensor values are not forwarded to the expert system itself, but are first fed to the Data Handling generic enabler. The CEP engine transforms the raw data into knowledge (temperature high, humidity low, during the last half hour), and by using this high level information the expert system produces its advice.

Publish/Subscribe Context Broker

A short description about this Generic Enabler is provided in the Business and Legacy System Integration section (see section 4.4.1.2). In the present section we will concentrate on how it can be used within the context of the Internet of Things.

Publish and Subscribe Context Broker Generic Enabler can be particularly useful in simplifying the deployment of applications that want to make use of the internet of things, as it allows total decoupling between the producers and consumers, meaning that context information is consumed, without having to know which Context Producer has published a particular event: they are just interested in the event itself but not in who generated it.
The event mentioned here could be anything of interest to an application, from an EPC related event (e.g. RFID tag read) or a temperature reading exceeding some threshold.

As the NGSI10 operations dictate, there are three possible ways of exchanging information:

- One time queries
- Periodic subscription where the subscriber is notified periodically about the subscribed events.
- Subscription where the subscriber gets notified only if there is new information about his subscribed events.

This generic enabler could be used so that a FIspace app would register itself as a context consumer, while external systems on the other hand could be registered as context producers.

For example, an advice app could register itself as a context consumer, receiving notifications when there are new advices coming from an expert system. The expert system then would play the role of a context producer, producing advices, but it would on the same time be a consumer of sensor measurements, of a farm management system.

4.4.3.3 Potential Technology Choices

Besides the already presented Generic Enablers, in this section we will present some tools that we believe will help towards realizing the IoT System and Data integration.

Fosstrak

Fosstrak is an open source RFID platform that implements the EPC Network specifications from GS1. It consists of several modules that can be utilized by system integrators or application developers who want to process RFID tag reads. Specifically, the EPCIS repository module implementation of fosstrak provides, except from an EPCglobal-certified EPCIS Repository, clients implementing the query and capture interfaces(SOAP and HTTP bindings respectively), as well as an adapter that can provide EPCIS access in a RESTful way. These clients can be used as they are, perhaps modified to suit our needs, or they can serve as blueprints to our own implementations. Another interesting and potentially useful module of Fosstrak is a Tag Data Translation Engine that can perform transformations between various EPC representations or encodings such as Binary, tag-encoding URI, pure-identity URI, or other legacy formats.

jCoAP

JCoAP is a Java Library implementation of the Constrained Application Protocol (draft-08). Both the client and the server part are implemented, although in the context of T250 we are mostly interested in the client part, since the IoT devices hosting the resources will be located in the external systems that FIspace apps want access to. Some already supported features include a simple API and a reliable and unreliable messaging. A proxy implementation is also provided, that carries out HTTP-CoAP or CoAP-HTTP translations, besides the traditional proxy functionality. The advantage of having a proxy is that pure HTTP clients can access CoAP endpoints, without having to implement the CoAP protocol.

Californium

Californium is another CoAP Java implementation (draft-13). It focuses on backend services and thus not much attention has been given on resource efficiency for embedded devices, but rather on usability and features.

4.4.3.4 Release Plan

This section provides an initial release plan for the main features of the IoT integration.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
EPCIS standard integration	Interface provision for the support of EPCIS related information exchange	-	initial	final
NGSI 9/10 incorporation	Incorporate the NGSI interfaces provided by FIware	initial	refined	final
CoAP Client API provision	Provide an API so that Fispace apps can access information hosted directly on sensor nodes	-	initial	final
Tool supported mechanisms	Provide the set up and configuration of the Generic Enablers that will support the IoT integration process	initial	refined	final

Table 20: Release Plan Internet-of-Things System Integration

4.4.4 High-level Technical Architecture

After the introduction of the main features of T250 – System and Data Integration now the initial conceptual architecture of this component is presented. Figure 25 shows a graphical overview of the System and Data Integration component as well as its relationship with other FIspace modules and external systems in the form of a UML Component Diagram. As indicated in the figure, the conceptual architecture consists of three main subcomponents: *External Systems Interface Provisioning Module, External Systems Interface Implementation Instance* and *FIspace Standard API*. Those are interconnected to each other and also hold connections to other FIspace components, such as the B2B Collaboration Core or the Security, Privacy & Trust component. In the remainder of this section we describe each subcomponent in a separate subsection.

4.4.4.1.1 External Systems Interface Provisioning Module

Interface Pool Management – module responsible for managing the pool of standard interfaces and their implementations to access external systems. This module should e.g. allow to add/register new interfaces/implementations, which will then be available for apps to access legacy systems.

Interface Discovery & Binding – allows searching for existing interfaces and their implementations, selecting the relevant ones and integrating them into an app or core module (e.g. an app developer searches for interfaces to a specific FMIS, finds the respective interface and one or more implementations of that interface. He/she selects the interface and implementation he/she requires and the module will provide the necessary JAR files to include into the app).

4.4.4.2 External Systems Interface Implementation Instance

This component represents one actual implementation of one of the common interfaces, enabling access to a specific type of external system.

Data Transformation Module – takes care of transforming data from FIspace/App specific format to external system specific format and vice versa. This transformation process could incorporate data mediation tools for mapping between the data formats prescribed by FIspace and those supported by the external systems (see section 4.4.2).

External Systems Communication Module – handles the actual communication with the external system based on the specific communication protocol supported by the external system (e.g. HTTP, JMS, CORBA, etc.)

4.4.4.3 FIspace Standard API

Standard API Web Service – the FIspace standard API, planned to be exposed as a RESTful web service. The Web Service uses the Security, Privacy & Trust Component (T270) to authenticate and authorize calls from External Systems to the FIspace Standard API.

Publish/Subscribe Component – a module forwarding data/events from external systems to interested apps, allowing the apps to subscribe to specific types of data/events they are interested in. Furthermore, this module allows defining access rights to the published data/events via the Security, Privacy & Trust Component that is developed in T270.

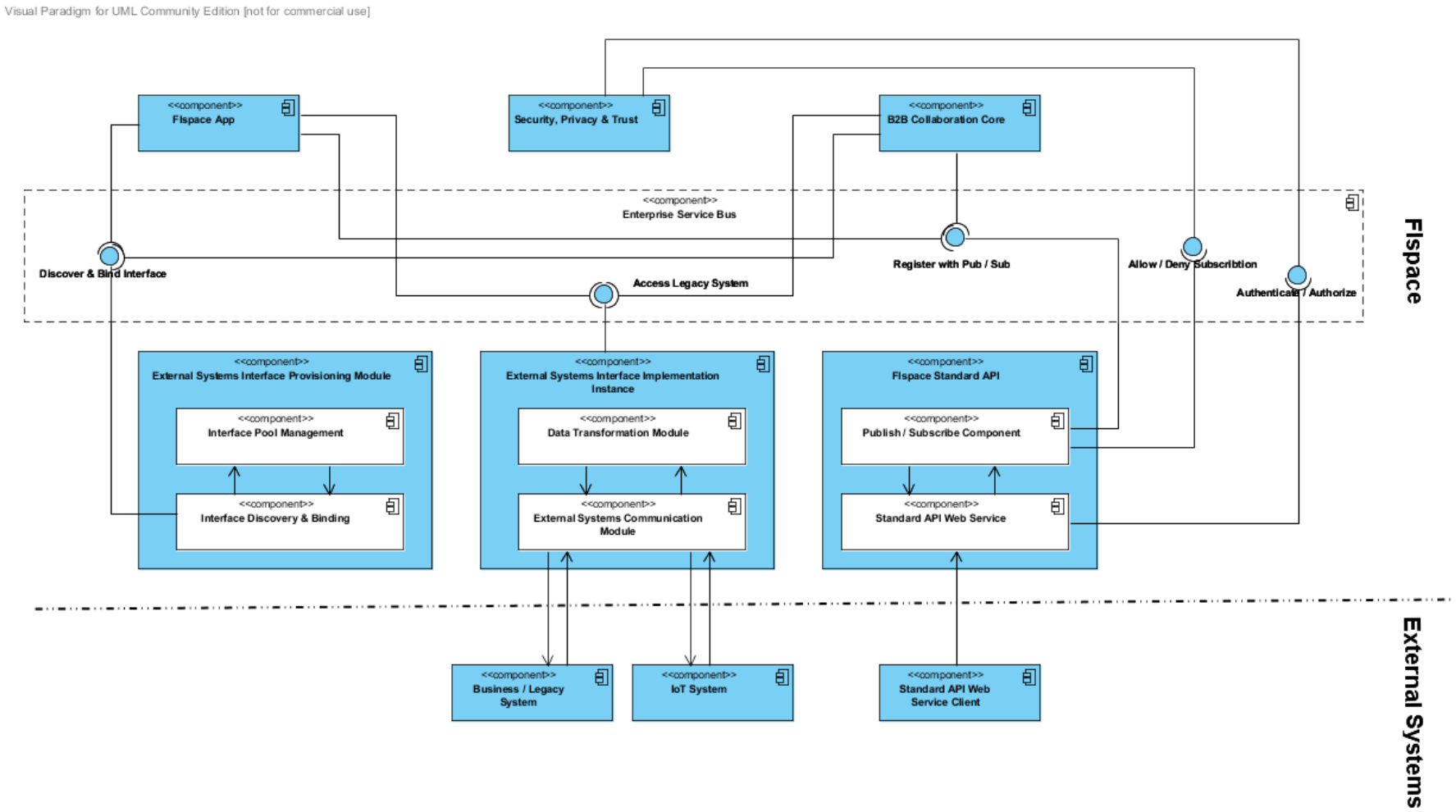


Figure 25: System and Data Integration – initial conceptual architecture

4.4.5 Overall Technological Choices

The system and data integration component consists of three sub components:

1. Service interface specification and domain model
2. Backend system Service provider implementations
3. Service provider registry

Modularity

Modularity and a modular runtime environment are essential for system and data integration. Service provider implementers should be able to deliver an isolated deployable module. They should specify thei2r dependencies in a deployment descriptor. The runtime environment should enforce code runtime isolation, so one service provider implementation cannot affect other service provider implementations or even worse FI-Space core components.

OSGi

OSGi is the most standardized specification for modularization in the java programming language. Roughly it consists of a runtime containers and modules. The modules are deployable in the runtime container. Each module has a description with it public packages and dependencies on other modules. The runtime container enforces module class loader isolation.

More about OSGi can be found online in the web *http://en.wikipedia.org/wiki/OSGi*

There are several OSGi framework implementations. Following list is limited to the open source variants:

1. Apache Felix

Apache Felix is an open source implementation of the OSGi Release 4 core framework specification. Felix is used in a number of bigger projects among which the Glassfish Java EE container and NetBeans IDE.

2. Concierge OSGi

Concierge is an OSGi (Open Service Gateway Initiative) R3 framework implementation intended for resource-constrained devices like mobile and embedded systems.

3. Equinox OSGi

Equinox is an Eclipse project that provides a certified implementation of the OSGi R4.x core framework specification. As such, Equinox is a module runtime that allows developers to implement an application as a set of "bundles" using common services and infrastructure.

4. Knopflerfish

Knopflerfish is a non-profit organization, developing OSGi related material. The project provides an easy to use open source certified implementation of the OSGi R4 v4.2 core framework specification, as well as related build tools and applications.

Of the two most used containers (Felix and Equinox), only Felix is designed as a standalone container. This makes it the preferred container. It also provides flexibility in the choice for overall FI-Space deployment structure. Since Felix is also embedded in GlassFish, the system and data integration component could be deployed on the same server as the GUI (which could be useful for testing, but probably not desirable in a production environment).

Distributed deployment

The system and data integration component could be deployed on multiple servers. This could be useful to provide local (geographically) runtime containers for service provider implementations. This will radically speed up communication between FI-Space and backend systems in most cases. The distributed runtime containers can be connected using DOSGi (where D stands for distributed). This functionality is provided by Apache CXF, which also include libraries to connect a variety of service interfaces (SOAP, REST etc.).

More about Apache CXF and more specific Distributed OSGi can be found in the web
http://cxf.apache.org/distributed-osgi.html

4.5 The Operating Environment

In this section we discuss the technical infrastructure that allows all of the FIspace components to work together in harmony. The main goal is to support the SOA principles on a Cloud that is FI-WARE based, to accommodate the expected scale the solution is distributed and allows “local” decision making with incomplete information.

The Operating Environment provides automation supporting the application lifecycle and support a “scale out” design model that is decentralized with redundancy for failure tolerance and auto recovery. It supports eventual consistency, as well as strong consistency asynchronous models. It will build on the FI-WARE GEs. It is planned to enable the distributed load balancing, elasticity and placement building on the FI-WARE GEs. This shall be augmented by close alignment with the security components (see Section 4.6 below). The team has very strong expertise in relevant areas of distributed systems, scalable communication infrastructure, monitoring and synchronization services, and its previous works are planned to serve as a basis for the FIspace Operating Environment development, in particular [13–21].

4.5.1 Overview & Main Features

The Operating Environment will provide the interaction medium for all FIspace components that serve as the basic building blocks of FIspace as well as the applications. The following aspects are supported by the Operating Environment:

- Management of the “composed service (application)” life-cycle, based on IaaS Cloud related OSS and BSS (planned to be provided by FI-WARE)
- Enterprise Service Bus that is based on Peer-to-Peer Overlay technology, supporting:
  - Eventual consistency
  - Events Bus, Management Logic
  - Pub/Sub Abstraction for information dissemination
  - A Bulletin Board abstraction for filtering and orchestration
  - Queues supporting various QoS for delivery and execution (e.g., once only, multiple readers,)
- Consistency and Replication Service. This service is partition tolerant and guarantees strong consistency (when needed)
- Operational registry for maintaining runtime attributes and supporting real-time operations
- Multi-tenancy support, with the least effort from the developers (both FIspace developers and then the applications developers)
- Monitor the KPIs and health, automate the operation, enforce the SLA, facilitate the problem determination, continuous optimizing the runtime

4.5.2 Interaction Protocols and Interfaces

The backbone of the FIspace Operating Environment is the Cloud Service Bus. Other components will need to interact with it using its interfaces and protocols. Regular operations, events and notification will be communicated through this bus. The Bus will also be used for communicating health monitoring of the applications and base services. We develop protocols to make sure that false positive and or false negative indications are minimized. One of the first jobs of the Task 260 team is to define these interfaces and protocols. We define a client library that will be part of the SDK and used by applications that interact with the base services and or other applications. We plan to also utilize REST interfaces and notification services, in alignment with what is available in FI-Ware.

4.5.3 Cloud Service Bus

The Cloud Service Bus (CSB) layer of FIspace Operating Environment is the middleware that enables integration of different FISpace service components and applications. CSB provides a rich set of integration interfaces and qualities of service, required to support a wide spectrum of information exchange scenarios, ranging from best effort notifications, to guaranteed delivery of transactional data. The Cloud Service Bus has virtually unlimited scalability, both in the number of supported end-points, and in the number of communication channels allowed by the Bus transport fabric.

The scalability challenge, posed by the large number of data producers, service providers and consumers in this cloud-based project, is addressed by Peer-to-Peer overlay technology. The CSB core is comprised of Bus Nodes, connected by SR structured overlay fabric [ref]. The CSB clients are the FIspace service components and applications, using the Bus for integration and connectivity. This two-tier architecture allows us to build an enterprise-strength service bus, with extensive functionality set and assured delivery of transactional data. The CBS supports multi-tenancy, enabling creation of separate Bus domains, each using a different P2P overlay structure.

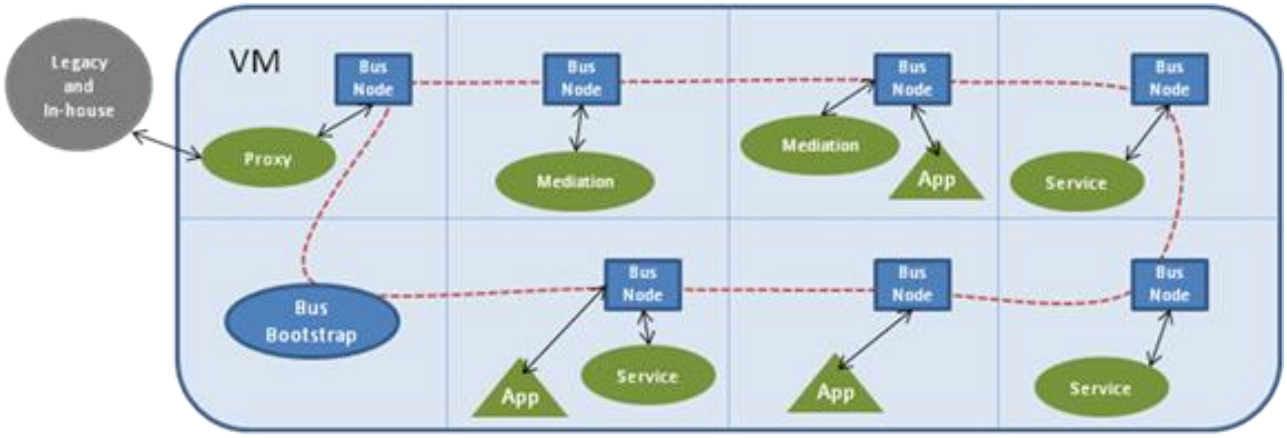


Figure 26: Illustration of FIspace Cloud Service Bus

The Cloud Service Bus will provide the following functionality:

4.5.3.1 Publish/subscribe

Topic-based publish/subscribe is an efficient tool for scalable message distribution to multiple clients. A topic is a virtual address, where any client can send (‘publish’) a message, and any client can listen (‘subscribe’) to the published messages. Publish/subscribe is optimal for fast and scalable distribution of real-time data. It provides best effort or partially reliable qualities of service. The mode of data delivery is synchronous - receivers need to be online when a message is published, and have to keep up with the transmission rate.

4.5.3.2 Queueing

Unlike Topics, Queues are physical objects, with an associated persistent storage on a hard disk. Any client can send a message to a queue, where it will be written to a disk and replicated for high availability. Any client can consume the queued messages. Queuing is the most reliable and consistent data delivery method, supporting persistence, identical message ordering and high availability qualities of service. Also, queues allow for fully asynchronous data exchange – much like in email, receivers don’t have to be online when messages are sent; they can join a queue any time later and retrieve the stored messages. These qualities of service come at the expense of performance – queuing has lower throughput and higher latency, compared to publish/subscribe.

4.5.3.3 Membership query and notification

The Bus Nodes are members in the peer-to-peer overlay structure. Every Node keeps the basic information on all other member Nodes, and on all clients attached to them. CBS will provide an application with means to tap into the client membership information – query the client description, topic and queue interest, get notifications on client start and stop events, etc.

4.5.3.4 Direct data link

CBS allows applications to create direct client-to-client links, for unbrokered delivery of bulk data. This might be useful in certain one-off situations, e.g. when a large file needs to be sent from one application to another. No CBS qualities of service, such as reliability or scalability, are available in this mode, and a number of allowed direct links per client is restricted.

4.5.3.5 Service mediation

Cloud Service Bus, in a later version, will provide an additional layer of functionality, required for service and application integration support, available in typical Enterprise Service Bus products. This layer will cover service request/reply interface, support for REST/HTTP , SOAP, legacy protocols, data format transformation, custom routing and other service mediation types. This will be done in close cooperation with the Task 250 team, who work on interoperability with in-house data legacy protocols and message format translation. We will create an integrated architecture, tying all components together in order to define a fully functional Enterprise Service Bus for reliable and scalable integration of all FISpace services and applications, running both on and off premises.

With respect to the above feature definition, the table below shows the initial release plan.

Table 21: Release Plan for FIspace Cloud Service Bus

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Pub/sub and Queuing	Interface and implementation of pub/sub and queuing services	beta	Full version	Bug fixes
Membership and direct link	Interface and implementation of membership and direct link services	-	beta	Final version
Mediation support	Definition and implementation of unified ESB architecture	-	-	Final version

4.5.4 Service Bus Monitoring

A monitoring agent available in each active node (VM - virtual machine) will collect health information about the applications and the basic components that comprise FIspace. This monitoring agents will have limited decisions capabilities (based on the monitored information) and will interconnect through the CSB to other monitoring agents. They will also participate in data collection for reports generation. An agent is also connected to the applications that run on the particular VM. This connection is to a software component that is part of the application and is supplied by the SDK with the set of KPIs that the application requests to monitor.

With respect to the above feature definition, the table below shows the initial release plan.

Table 22: Release Plan for FIspace CSB Monitoring

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Monitoring Agent	Monitoring agent that collect information about the application's list	-	beta	Full version
Monitoring for the SDK Library	Health monitoring software that is part of the application	Concept prototype	Working version	Bug Fixes

4.5.5 Consistency Services

This important component is the base for operations that require consistency. Starting from maintaining configuration records through an active operational registry and ending with supporting ACID transactions. Tlaloc is a new Paxos based, fast consistency service platform aiming at improving elasticity of consistency services for applications hosted on FIspace or FIspace components themselves. In the core of Tlaloc is a novel replicated state machine protocol, which employs speculative executions to ensure continuous operation during the reconfiguration periods as well as in situations where failures prevent the agreement on the next stable configuration from being reached in a timely fashion. The Tlaloc platform could be used for many state replication application in clouds such as leader election, distributed locking, application state replication, lock service, data replication, etc.

With respect to the above feature definition, the table below shows the initial release plan.

Table 23: Release Plan for FIspace Consistency Services

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
Base Consistency service	Paxos based cohort in the cloud that supports consistency services	beta	Final version	Bug Fixes
Operational Registry	Keep vital information about application's configuration and dependencies	Concept prototype	beta	Final version
Replication service	Support the high availability of application	Concept Prototype	beta	Final version

4.5.6 Generic Enablers and Technology Choice

As the baseline for commencing the implementation, the following technology choices were made:

1. Using a p2p overlay technology as a basis for the CBS and monitoring.
2. Using agent based technology for monitoring system's and application's health as well as reporting
3. Speculative Paxos based technology for maintaining consistency wherever it is needed.

In addition, we are investigating the Generic Enablers currently developed in FIWARE as possible candidates for the operating environment implementation. The following table shows an initial validation of the availability and usability, which will be further evaluated in upcoming milestones.

GE	Function	Where to use	Availability	Initial Evaluation
Catalogue	Used to display a catalogue of available FI-ware GEs	Allow Application developers view the available platform services	Available	Looks like it is designed more as a web interface and not as a development tool
Advanced Middleware	ESB	CSB	They just started working on it.	DDS looks like an overkill for the purpose of an ESB
Mediator	Adapting between protocols to enable the creation of composite services	CSB	Ready	We need to see if it is required by our basic services or the use cases.

Table 24: Generic Enablers for FIspace Operating Operating (initial validation)

4.6 Security, Privacy, and Trust

In this section we describe the Security, Privacy and Trust (SPT) features and overall design for FIspace based on a first analysis of industrial user needs. We identify the required SPT technologies, describe the initial implementation plan that to a large extend builds upon on the Generic Enablers provided within the Security Chapter of FIWARE as well as other well-proven technologies from Cloud Security Architecture and other frameworks for ensuring secure systems for enterprise.

4.6.1 Overview & Main Features

The aim of the SPT framework for the FIspace platform is to provide secure and reliable exchange of confidential business information and transactions using secure authentication and authorization methods that meet required levels of security assurance. SPT technologies from FI-WARE will provide security mechanism and assurance methods. FIspace Authentication, Authorization and Accounting technologies will provide user management & access control features. A Developer Framework is supported by SPT technologies to provide appropriate security solutions to the developers.

The main features of the SPT framework have been driven by an initial analysis of the SPT functionalities that will be required by industrial actors that will be users of the FIspace platform, and industrial technology suppliers who will exploit the FIspace platform to provide Apps and associated services to the industrial actors. The main feature categories that have been considered in the initial design of the SPT framework for FIspace are the following:

- Identity and Trust
- Authentication
- Access Control

A short summary of these features within FIspace are described in the following sections. In addition, we describe the security mechanisms that we intend to implement.

4.6.1.1 Identity and Trust

In early FIspace application scenarios, two actors establishing identity and trust to exchange information between their devices will often have some previous knowledge of one another having been in physical communication. The actors will therefore use a “biometric” comparison and validation as a basis for exchanging credentials for authentication. In more advanced and eventually more common scenarios, actors will not be able to rely on having physical contact with other FIspace actors.

Features

a) Actors may be only known by their online profile and possibly rankings or references of other actors. FIspace will need to provide mechanism for trust, without actors having any prior knowledge, that enables actors to validate credentials. FIspace will likely need to support multiple mechanisms for establishing initial identity and trust between actors involving third party credential providers. (e.g. mobile phone, credit card, national eID, etc.)
b) There will be actors/devices that need to be trusted that are not part of the FIspace user community. For example, we must be able to trust external sites providing status or data regarding business processes. More complex scenarios will likely involve trust of information coming from an external third party proprietary source. FIspace SPT framework will need to address how devices and third parties are integrated into the SPT policies and mechanisms and how information coming from these third parties is verified.
c) FIspace will need to provide a basis for verifying Company Administrators who a explicitly authorised by the company. In a small company this might simply be the person that has purchased the App and completed a registration process. In a larger company where there might be multiple administrators from different divisions, some centralised FIspace process must be provided for granting administrator privileges on behalf of company entities.
d) In more diverse application scenarios involving multiple companies, some central FIspace mechanism for validating a company’s identity will be needed. It might be possible to use the EU-wide VAT registration database as a basis for verifying company identity.

e) Industrial Apps for FIspace will utilise a more elaborated initialisation procedure for verifying identity then mobile based user productivity Apps. In particular, the company identity will likely need to be verified via human evaluation of the registration data of the company. In addition, some human interactions may be needed to determine who in the company is authorised to grant administrator status to specific individual users or to change company profile information.

4.6.1.2 Authentication

Several different types of authentication will need to be supported to carry out simple scenarios of users exchanging information utilising FIspace Apps and communication facilities. In particular, FIspace needs to provide the following authentications for the information exchanges between the actors based on the initial scenario analysis:

Features

a) Authenticate information sent from individual users – users need to be sure requests from other users can be trusted (are worthwhile to respond to) and when users receive a response they need to be sure who sent the response (can rely upon) before taking actions.
b) Authenticate information from third party systems – users will utilise information stored on third party systems so mechanisms that authenticate the source of information from third parties are needed. Facilities to register and “wrap” third party information sources to utilise FIspace recognised authentication will be needed.
c) Authentication of networked resources – the application scenarios supported by FIspace will integrate many different types of automated data sources such as sensor networks, e-tag scanners, positioning monitors, etc. Authentication of data as originating from expected sources will be needed in order that data be relied upon for business decisions.
d) Messages and event notifications should be encrypted and authenticated throughout the FIspace system to ensure FIspace only enables private communication channels between actors and within application and user domains.
e) Data objects such as reports, files and other business process artefacts need to be easily moved and hosted in a range of storage facilities locally and cloud based and must be protected from unauthorised access. Authentication and access control mechanisms must allow data objects to be self-contained, easily moved between locations and devices, and remotely authenticated.
f) A FIspace design decision will be whether to authenticate devices a user is using, or to authenticate the user regardless of device or some combination. Apps will likely carry out automated tasks within FIspace such as generating events notifications based on monitoring of activities or data. FIspace needs to authenticate such notifications even if there is no explicit user involvement or associated user authentication. There are also embedded devices such as scanners used in logistics where the user of the device may not be known as a registered FIspace user.

4.6.1.3 Access Control

Once two actors using FIspace have validated their trust in one another’s identity, the areas where access control features are needed within the FIspace framework are the following:

- Access to FIspace communication channels for messages, event notifications services and other system wide FIspace resources
- Access to protected user data objects such as reports, planning files, and business process data files.
- Access to protected user networked resources such as sensor networks, databases
- Access to protected third party resources such as government sites or commercial services
- Access to user attributes and person profile data
- Access to specific App components or App features based on type or level of user

As the user scenarios are further elaborated it’s expected that additional access control features will be required throughout the FIspace system and for a wide range of actions.

Features

a) We need to utilise a federated access control mechanism in order that domain specific access privileges are managed locally, while still enabling collaboration and granting of access between actors across different domains.
b) A key feature related to access control will be revocation where we will need to support real-time revocations in support of dynamic business relationships and in response to security threats.
c) As an additional security mechanism for commercially sensitive information, it may be necessary to place information under access control in a protected space on a FIspace node or the Cloud to ensure only information explicitly identified as authorised for sharing by a user is accessible and to minimise vulnerabilities to malicious attacks.
d) FIspace will need to integrate access control methods with third party access mechanisms such as username/password to control access to information sources external to FIspace such as government and commercial sites, as well as to legacy systems.
e) As FIspace is intended to be deployed within a broad multi-company community, the notion of domains of users (e.g. groups or company personnel) and domain (or company) membership will need to be supported. Some substantial verification of administrator identities will be needed as they will likely have privileges to determine which individuals are members of a group (e.g. company domain).

4.6.1.4 Threat Analysis

The Cloud Computing approach that will be utilized by FIspace represents a significant shift in information technology for actors within the Food and Logistics sectors. Reaching the point where computing functions as a utility has great potential, promising opportunities for new innovation in Apps and services. However, there are risks of Cloud Computing if not properly secured, and the loss of direct control over systems for which they are nonetheless accountable. SPT design in FIspace will focus on awareness and protection from cloud threats as explained in Table 25.

Threat	Description	Example	Remediation
Data Loss or Leakage	Deletion or alteration of records or loss of encoding keys may lead to loss of data	Insufficient authentication, authorization, and audit (AAA) controls; inconsistent use of encryption and software keys; operational failures; persistence and eminence challenges: disposal challenges; risk of association; jurisdiction and political issues; data center reliability; and disaster recovery.	Strong API Encrypt data in transit Data Protection Strong Key Management Data Storage
Account or Service Hijacking	Attack methods such as phishing, fraud and exploitation of software vulnerabilities still achieve results. Credentials and passwords are often reused, which amplifies the impact of such attacks.	NA	Prohibit the sharing of account credentials between users and services. Leverage strong two-factor authentication techniques where possible. Employ proactive monitoring to detect unauthorized activity.
Insecure Interfaces and APIs	The security and availability of general cloud services is dependent upon the security of these basic APIs. From authentication and access control to encryption and activity monitoring, these interfaces must be designed to protect against both accidental and malicious attempts to circumvent policy.	Anonymous access and/or reusable tokens or passwords, clear-text authentication or transmission of content, inflexible access controls or improper authorizations, limited monitoring and logging capabilities. Unknown service or API dependencies.	1-Analyze the security model of cloud provider interfaces. 2-Ensure strong authentication and access controls are implemented in concert with encrypted transmission. 3-Understand the dependency chain associated with the API

Threat	Description	Example	Remediation
Abuse and Nefarious Use of Cloud Computing	Spammers, malicious code authors, and other criminals have been able to conduct their activities with relative impunity. PaaS providers have traditionally suffered most from this kind of attacks; however, recent evidence shows that hackers have begun to target IaaS vendors as well. Future areas of concern include password and key cracking, DDOS, launching dynamic attack points, hosting malicious data, botnet command and control, building rainbow tables, and CAPTCHA solving farms.	IaaS offerings have hosted the Zeus botnet, InfoStealer trojan horses and download for Microsoft Office and Adobe PDF exploits. Additionally, botnets have used IaaS servers for command and control functions. Spam continues to be a problem as a defensive measure, entire blocks of IaaS network addresses have been publicly blacklist	Stricter initial registration and validation processes. Enhanced credit card fraud monitoring and coordination. Comprehensive introspection of customer network traffic. Monitoring public blacklists for one's own network blocks.

Table 25: FIspace cloud security threats and remediation

4.6.1.5 Security Assurance

In order to achieve acceptance and adoption by industrial users, FIspace must provide strong security assurance that commercial information and transactions are secure, can be trusted and are not vulnerable to malicious actions. FIspace will use a compositional security assurance process, separating concerns where possible. This has the added advantage that assurance data may be re-used for multiple FIspace instantiations. By re-using data we can considerably cut down on costs (for re-analysis) and also time taken to develop and deploy the FIspace system (security assurance analyses may be time consuming to perform). In a component based design process, independently developed components are assessed and matched to specific system security requirements to determine if they meet the system security objectives. For independently developed components such as Apps it is possible to provide assurance provided we can verify an App adheres to a set of system-wide and App-specific security policies. As the cost of full verification of independent Apps is costly and time consuming, FIspace complements the verification of security policy adherence by Apps with monitoring mechanisms to detect and prevent unacceptable or unexpected App behaviour. For example, FIspace may specify that an App must always utilise encrypted messaging, which can be monitored and verified during FIspace runtime.

As part of the compositional assurance process, it's possible to assess if there is evidence to convince commercial users (or a certification body) that the FIspace system level security requirements are adequately met by the FIspace implementation. We take required security properties, and their assurance, and match them to system level assurance requirements. This is summarised in Figure 27.

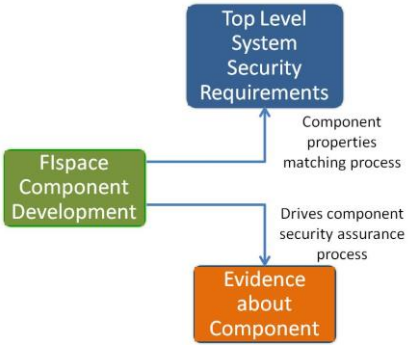


Figure 27: Overview of compositional security assurance process

The FIspace security policy architecture is used to describe individual, permitted interactions between components, Apps, and users. The FIspace platform will enforce and respect the security policy architecture. Rather than build an individual monolithic assurance case for FIspace, we wish to establish fundamental security principles, and build an assurance case for various App and industrial scenarios, re-using assurance data about the FIspace platform elements. The relationship between these elements is shown Figure 28.

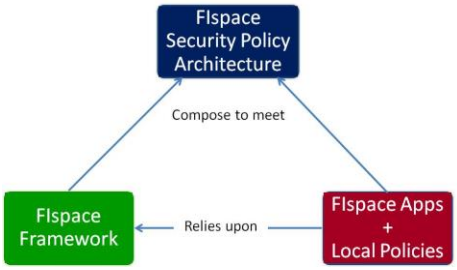


Figure 28: Elements of FIspace security assurance

The top level system security requirements that satisfy required security assurance levels have been taken into account in advance of more detailed specifications of the application scenarios being defined in WP400. An initial set of security policies have been defined for the following SPT functions:

- Data object storage
- Access control
- Attribute based control
- App authentication
- Data object authentication
- Information flow control
- Internal transfer protection
- Access to third party data sources

The policies in each of these areas are the drivers behind the initial SPT technical architecture described in Section 864.6.2. These will be expanded in number and detail as further information concerning the scenarios and user requirements from WP400 are more fully defined. A summary of the initial policies established for each SPT function is provided in Table 26.

Security Function	**Security Policies**
Data object storage	All data objects (e.g. electronic documents, reports, application data files, etc.) will be stored in encrypted format and bound with meta data usable for enforcing access control policies.
Access control	FIspace shall enforce a set of access control policies on all stored data objects, assignment and modifications of user attributes, and access to event notifications. FIspace shall ensure that all operations between any FIspace user and any data object are covered by an access control policy.
Attribute based control	FIspace shall enforce access control policies for data objects based on user groups such as administrator, user, company representative, and for data objects classifications such as confidential, company confidential, and amongst named users and groups of users.
App authentication	FIspace shall provide a capability to generate evidence that can be used as a guarantee of the validity of messages and event notifications as originating from registered FIspace Apps.
Data object authentication	FIspace shall provide data objects with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.

Security Function	Security Policies
Information flow control	FIspace shall ensure all user level information that flows through the system (e.g. messages, event notifications) is under access control policies. FIspace shall ensure that all operations that cause any information to flow to and from any user are covered by an information flow control policy.
Internal transfer protection	FIspace shall enforce access control policies and information flow control policies to prevent the disclosure or modification of user data objects when transmitted between physically separated parts of the FIspace framework.
Access to third party data sources	FIspace shall utilise component authentication mechanisms when importing data objects from outside of the FIspace framework to generate evidence that outside data objects were introduced by authenticated FIspace components.

Table 26: Initial FIspace security functions and policies

The security functions and policy enforcement are implemented by the components that comprise the technical security architecture described in the following section.

4.6.2 High-level Technical Architecture

FIspace High-Level Technical Architecture is mainly based on FI-WARE security solutions and security assurance process.

4.6.2.1 Security Architecture

FIspace security architecture relies on the identity and trust relationship between users and applications based on roles and access rights with data protection provided at rest or in flight. Incidents will be monitored and transfer of malware infected files will not be allowed. Logs from various modules and service provider solutions will be collected and analyzed to provide Security Incident Management. Strong Authentication, Authorization with the minimum access rights and control mechanisms are the key concerns of the security architecture of FIspace. Encryption of data in rest or in motion with access control mechanisms will provide secure and reliable architecture for the commercial data. Controlling and sharing public data will be an integral part of the user’s business network.

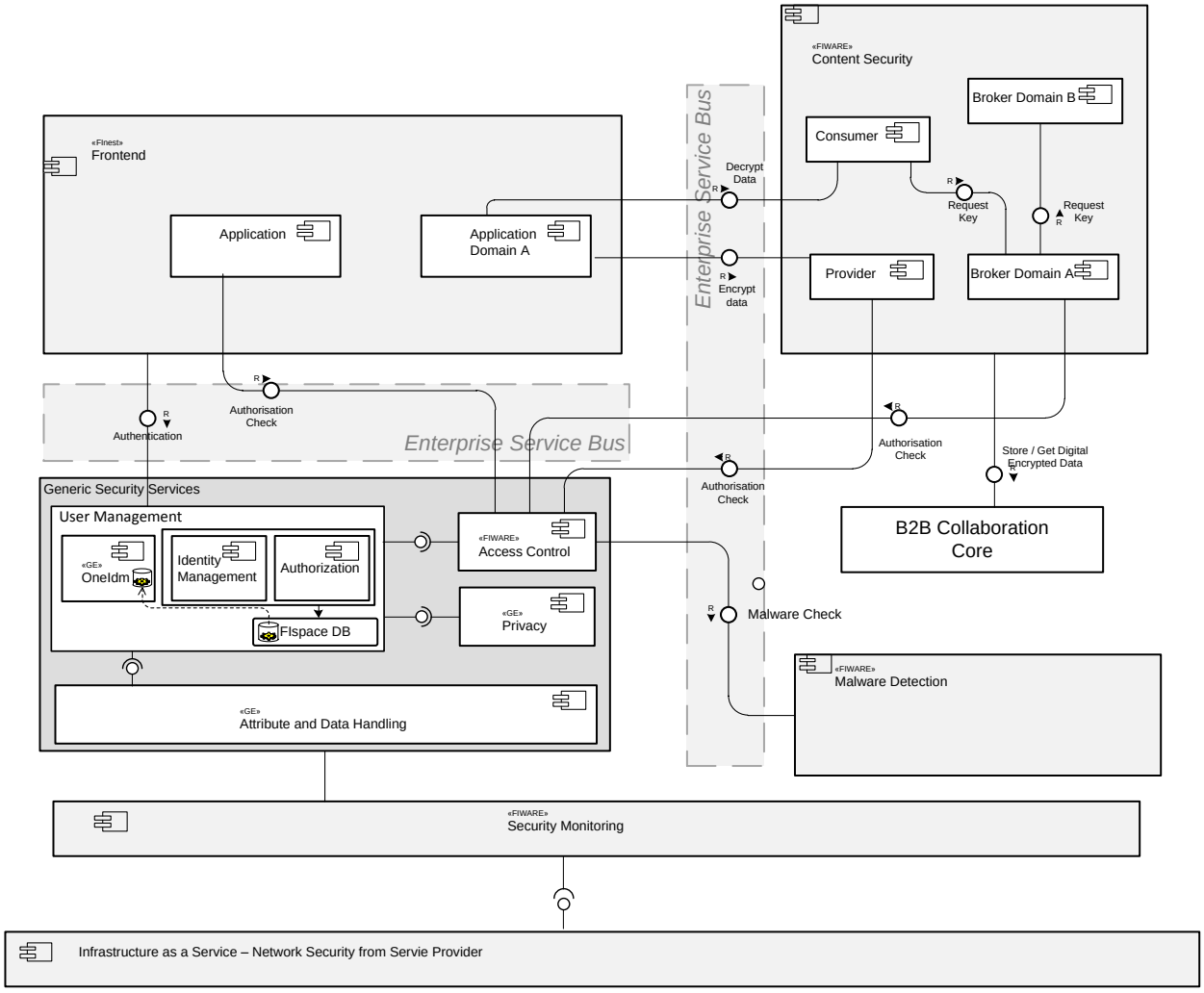


Figure 29: Security-Privacy-Trust Conceptual Architecture

4.6.2.2 Identity Trust and Access Management

Conceptual design for generic security services consists of four Generic Enablers (GEs) (including OneIdM, Privacy, DataHandling and Access Control), LDAP server (authorization and IdM components) and a GE independent database.

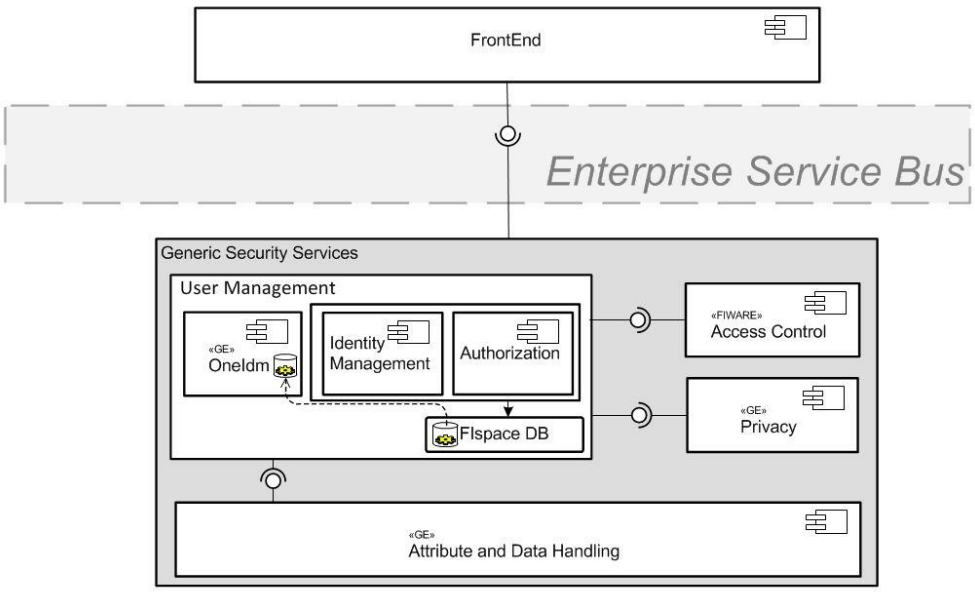


Figure 30: FIspace Security Layer

The Security Services layer as shown in Figure 30is designed to control authentication, authorization and identity management as well as user privacy and data security. User Management Agent contains OneIdM GE and LDAP technologies. Initial design was to keep authentication, authorization and identity management under OneIdM GE. However, due to limitations of the OneIdM GE, the architecture needed to be redesigned. OneIdM GE is still used, but it can only be used as an authentication system. Therefore, the architecture is redesigned to give authorization and identity management functionality.

Based on our design, an LDAP server will be used for identity management and security policies of LDAP’s group attributes for role-based access control. User account creation and managing personal information is controlled by OneIdM GE. Since, FIspace system requires user hierarchy and role-based access to allow users to access certain apps, information or services; user hierarchies are used to organize relationships between the companies, authorized person and regular users. Hierarchies also permit users to access certain information that all the company users should be able to access. Users also require having some user specific access such as user’s personal information or to an application only a user can access. This functionality will be delivered by defining security policies to LDAP’s group attributes.

Authorization (RBAC) and identity management information will be kept on a secure database in the FIspace system, and this database will be kept synchronized to OneIdM user database. For every user account created in the OneIdM GE, an equivalent user will be created in the FIspace database. All the personal information is going to be preserved in the OneIdM DB, and other information is going to be kept in the FIspace DB.

The Privacy GE is used in this system to offer trustworthy, yet privacy-friendly authentication, using privacy-enhanced attribute-based credentials.

The Data Handling GE is a privacy-friendly attribute-based access control system, which targets mainly sensitive data. It permits to store information together with an attached privacy policy, which regulates its usage.

Detailed information about GEs can be found under Section 4.6.3 Generic Enablers.

4.6.2.3 Content Security

Data protection and access control is very important for the cloud infrastructures. Data in FIspace will be kept in encrypted form wherever stored or sent through internet. Data is protected and encrypted by provider and user access will be verified by Access Control / IDM depending on whether user has rights to access the data. Similarly, an application can see the data by using the consumer and broker to remove protections and again user access is controlled on secure access and IDM. Different Brokers can be used for different domains or customers, roles and access can be defined between them to access data.

4.6.2.4 Malware Detection

Uploaded binary files should be verified against to the threats or malware. Any application uploads of binary files will be sent to the malware detection GE. In case of detection of infected files, binaries will not be sent to FIspace modules.

4.6.2.5 Incident Management

Incident management is important to understand and analyze the FIspace infrastructure by collecting the logs from FI-WARE GEs and service provider’s network security solutions like firewall, ips, etc.

The main concerns of Incident Management are:

1. Normalize and correlate events, rise alarms
2. Collect the vulnerabilities and evaluate the potential threats
3. Identify attacks and score essential elements impact
4. Assess risk and propose remediation solutions
5. Deliver a user-oriented security visualization service allowing efficient monitoring from the security perspective.

4.6.2.6 SPT of Software Developer Kit

FIspace SDK is expected to be a well-defined and version controlled set of libraries and services that has been specifically assembled to meet the needs of App Developers but also to ensure security requirements are enforced. We expect the SDK will enforce security by requiring that security elements be utilized for most tasks. All Apps must use the same well-defined SDK as this is the only model that would provide the level of trust needed for businesses to carry out commercial transactions using FIspace**.**

Security is to be designed into the system using a well-defined set of security protocols for implementing each of the security features.

APIs provided by the SDK impose compliance with the Security Architecture and security policies. Built-in security features provided by the FIspace SDK will be the following:

- **Authentication**: Identity verification to verify whether a user is legitimate or not.
- **Access control:** Verification of access to data objects, Applications, Application components, third party data sources and devices/networks.
- **Data handling**: Intended access to the data by the application should matches the scope defined in the usage policy provided by the user.
- **Data in transit**: Send messages and data objects to servers securely to protect the integrity and secrecy of data in transit.
- **Secure Access**: Store data securely with encryption on the cloud to protect data at rest.
- **Malware Detection:** Treat untrusted files and data with care.

Security is to be designed into the application or service from the very beginning, and make it a conscious part of the entire process from design through implementation, testing, and release. Security is designed into the system and is not an add-on or optional. In addition to security features provided by FIspace SDK, SPT developer guidelines will also be provided for the SPT framework. SPT developer guidelines consist of set of procedures and best practices, provided to increase security of FIspace applications, which includes secure coding techniques to **avoid exploitable coding flaws.**

4.6.2.6.1 SPT SDK Architecture

The FIspace Development environment as described below in Section 4.7 consists of a SDK based on Eclipse plugins. Eclipse is built-in using OSGI technology. OSGI is a set of specifications that define a dynamic component system for Java. The OSGI specifications enable components to hide their implementations from other components while communicating through services, which are objects that are specifically shared between components.

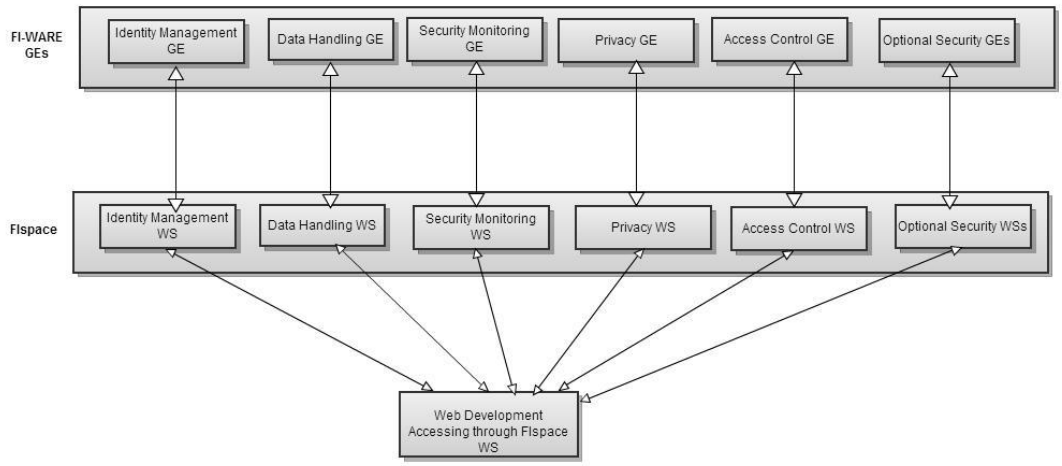


Figure 31: Concept of the SPT integration into the FIspace Development Environment

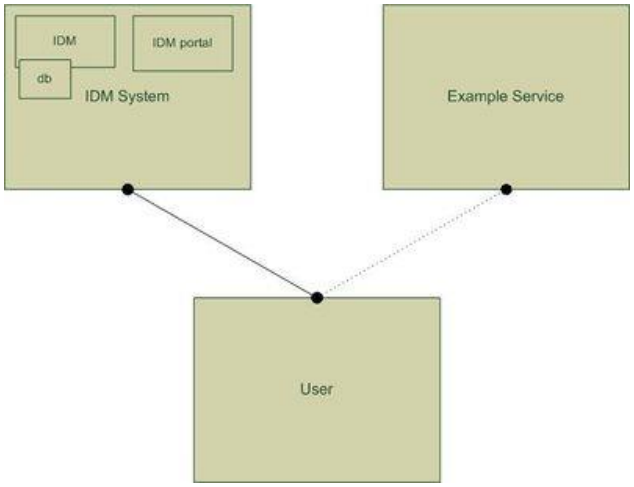
FIspace SPT SDK is one of the modules that compose FIspace plugin and it consists of sub-modules. Each sub-module simplifies API of FI-WARE SPT GEs and is responsible for providing libraries for SPT features provided by FIspace. Libraries will be implemented as wrappers of REST APIs defined in the FI-Ware Open Specifications.

4.6.3 Generic Enablers

SPT technologies from FI-WARE and necessary technologies from third parties will provide a secure and reliable design of FIspace. Planned GEs from FI-WARE are described in the following sections.

4.6.3.1 OneIDM GE

The OneIdM offers tools to reduce the effort for user account creation and management, as it supports the enforcement of policies and procedures for user registration, user profile management and the modification of user accounts. Administrators can quickly configure customized pages for the inclusion of different authentication providers, registration of tenant applications with access to user profile data and the handling of error notifications. For end users, the IdM provides a convenient solution for registering with applications since it gives them a means to re-use attributes like address, email or others, thus allowing an easy and convenient management of profile information. Furthermore, as it is possible to configure several applications that shall be linked to OneIdM, the main benefit for users is a single sign-on (SSO) to all these applications.

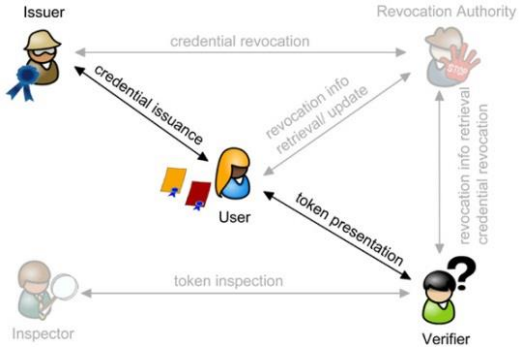


Initially, OneIdM GE is designed to provide Identity Management (IdM), authentication and RBAC (Role-based access control). However, due to some unidentified problems, OneIdM cancelled the development of RBAC, and IdM is lacking these major functionalities in its current state. Basically OneIdM GE only works as an authentication system. Therefore, for the shortcomings, alternative solutions must be created. After careful investigation, FIspace team decided to use LDAP to deliver IdM and an RBAC like environment. Detailed information can be found under 4.6.4.2 (Identity Management and Authorization).

4.6.3.2 Privacy

The Privacy enabler provides trustworthy, yet privacy-friendly authentication, using privacy-enhanced attribute-based credentials (Privacy-ABCs). Briefly, the User first obtains credentials, which are certified attribute-value pairs, from an Issuer who vouches for the correctness of the certified attributes. The User can subsequently authenticate towards a Verifier by sending a presentation token, which is derived from her credentials. A single presentation token can selectively reveal attribute values from one or more credentials.

For an easy integration of Privacy-ABCs in various applications and systems, a mechanism-independent ABC Engine layer on top of the core Cryptographic Engines is considered. This ABC Engine layer contains all the mechanism-agnostic components of a Privacy-ABC system. The figures provided in the following sections depict the high-level architecture around the Privacy GE and the typical communication flows for credential issuance and token presentation.

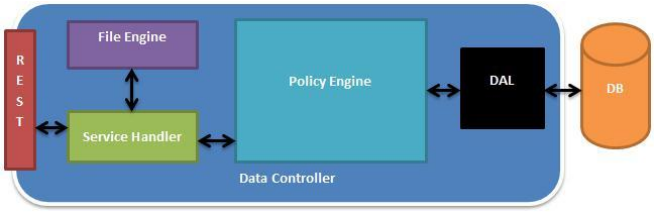


Grayed-out entities are optional

Privacy GE is planned to be published by the end of July. However, if it cannot be published; alternatively user’s hashed MAC address can be stored in a secure database to control “if the signed-on user is the actual user who initially entered the credentials”. This can be done by comparing MAC address stored in the database against the MAC address that user’s device is providing. If there is a mismatch, user’s session is terminated and login information is asked again.

4.6.3.3 Data Handling

The Data Handling GE is a privacy-friendly attribute-based access control system, which targets mainly sensitive data. It permits to store information together with an attached privacy policy, which regulates its usage. Thus, the Data Handling GE can reveal certain attributes, according to specific supplied conditions.

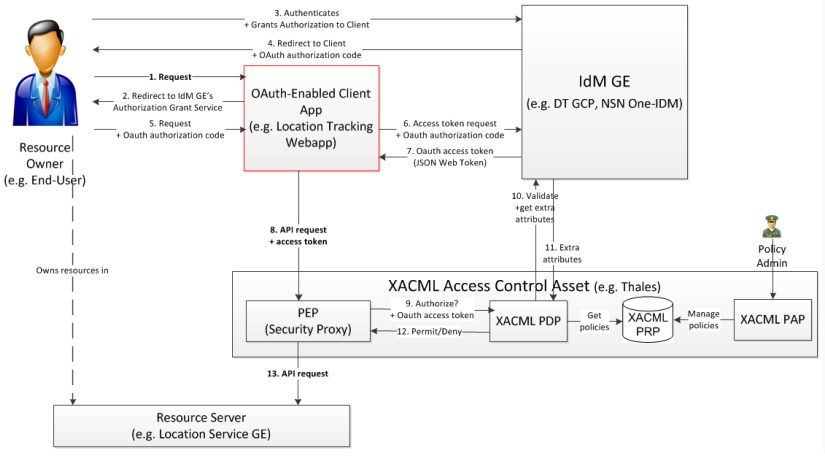


The Data Handling GE provides a mechanism for controlling the usage of attributes and data (more precisely, of Personal Identifiable Information or PII) based on the concept of ‘sticking’ a data usage policy to the data to which it applies. When the data is accessed by an application, an access control technology is then used to verify that the intended use of the data matches the scope defined in the usage policy. Therefore, the Data Handling GE can be used by any application or service that would offer a transparent data handling policy to users and third parties. In the example scenario later proposed, we propose to use the API of the Data Handling GE for a social network web site that is collecting private data of the subscribed users.

4.6.3.4 Access control

Access Control GE provides a mean for controlling access to resources that are available on a given FI-WARE Instance. Those resources are owned by users of the FI-WARE Instance, either end-users (data that belongs to each end-user) or by application or service providers (API operations exported by a given application or service, which may well be a FI-WARE GE). Such access will be typically requested by client applications or services (including FI-WARE GEs) acting on behalf of another user. It would require approval from the resource owner and may be restricted by security policies that are either global to the FI-WARE Instance, or defined for application/services or for the end-users (both resource owners and end-user on behalf of whom access is requested), as well as the organizations that end-users belong to, if any.

On the figure below, Service GEs provide API resources to Client Applications. Some of these APIs provide access to specific resources of some users. The IdM GE (Identity Management Generic Enabler) provides identity management and authentication for users and client applications.



Access Control and policies will provide flexibility to control user / groups access rights to use applications of FIspace and necessary FIspace modules and interactions between them. Policies should be well defined for users on XACML Access Control Asset Policy repository.

4.6.3.5 Content Security

The Content-Based Security Optional Generic Enabler provides protection at the application layer of items of data with channel independent and protection at rest or in flight.

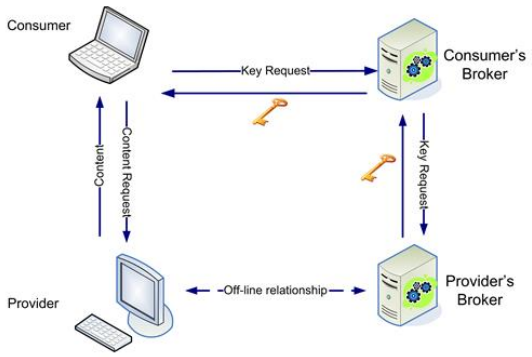
The CBS GE consists of three key components:

Producer - creates digital containers by adding protection to data by applying an encryption operation and/ or attaching a digital signature. The digital container can be stored or sent with no restrictions. The content is still protected and the key must be retrieved from the broker. The producer registers with a Broker in order to share keys with it, so that the broker can regenerate the decryption keys for each item.

Consumer – This component is used whenever access to the data is needed. It requests the decryption keys from the Broker and extracts data from Digital Containers by removing protection from data by applying a decryption operation and/or verifying an attached digital signature.

Broker – generates keys required to decrypt the data inside Digital Containers. The Broker uses the Access Control GE to check the user’s credentials and the information about the data object against a set of policies of use. In the architecture shown, the producer and consumer

In the architecture shown, the producer and consumer hare a broker. However, it is possible to support a network of broker components, rather than just one trusted by all the providers and consumers.



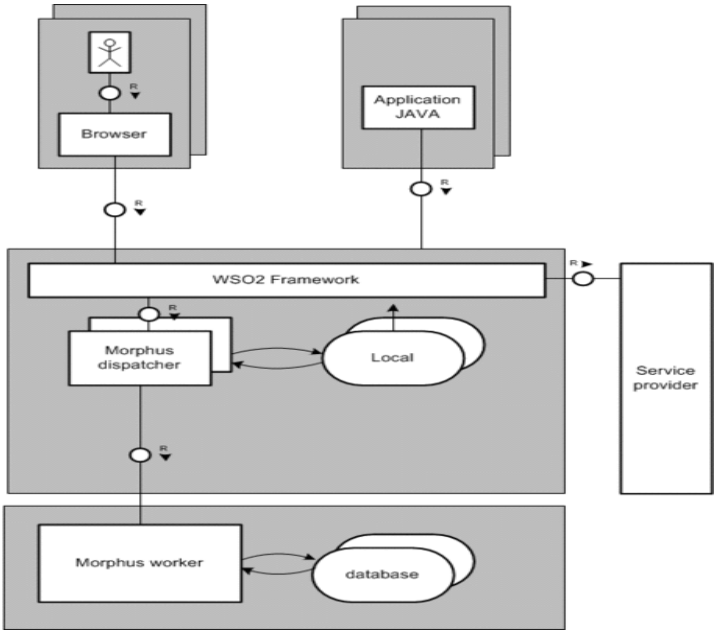
The CBS OGE has two main interfaces. The “protect” interface applies protection to data. The content producing application provides raw data and metadata. The CBS OGE returns the protected data in a digital container. The unprotect interface removes protection. The content consuming application provides the protected container and either its own credentials or those of the end user. The CBS OGE returns the raw data.

4.6.3.6 Malware Detection

The malware detection service GE provides a mechanism for determining if the submitted executable binary file is sane or infected by a malware. The Malware detection GE is available as a web-service or through a direct connection to the website and it can reply two different type of requests of the client for the submitted file.

- Scan: Gets binary file as an input and returns SANE/INFECTED string
- Distance: Gets binary file as an input and returns distance matrix to a malware database.This distance evaluation indicates the distance of the input sample with respect to the malware of the database

Malware Detection GE provides two options of analysis, static or dynamic. A static analysis is fast but not very precise; a dynamic analysis is finer but less efficient because binary must be executed in a monitored environment.



4.6.3.7 Context Based Security Compliance

Context-based Security & Compliance GE (CBS&C GE) provides the security layer of FI-WARE with context-aware capabilities to support additional security requirements through the optional security enablers developed in FI-WARE not provided by the generic FI-WARE security services.

4.6.3.7.1 PRRS Framework

PRRS (Platform for Runtime Reconfiguration of Security) Framework provides run-time support to end-users and client applications for performing dynamic selection & deployment of optional security enablers. PRRS framework is in charge of controlling the rest of the components of the GE, processing requests from end-user applications and orchestrating the deployment of the optional security enablers selected.

4.6.3.7.2 Rule repository

This component will allow the generic enabler to store and manage compliance requirements and 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.

4.6.3.7.3 Context monitoring

Runtime context monitors are the components in charge of detecting 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 the PRRS Framework. In case of non-compliance detection the assigned event will be sent to the PRRS framework by the appointed monitor so that the framework could take the necessary recovering actions

Context Based Security Compliance can be used on FIspace based on the needs of use case scenarios. Since this GE is capable of the monitoring of end user context environments and capable of making a decision and action according to the rules, we can implement this GE based on the needs of the use cases scenarios of FIspace.

4.6.3.8 Security Monitoring

Service Providers should use Security Monitoring based on regulations, compliance and to provide incident management. The Security Monitoring GE is part of the overall Security Management System in FI-WARE and as such is part of FI-WARE instances and information about its reference implementation is published on the FI-WARE Catalogue. The target users are: FI-WARE Instances and Service Providers, applications from third parties and soon commercial facilities as City Council, Service Operators, Business and entrepreneurs.

The security monitoring enabler is composed of the following functionalities:

- **Normalization of heterogeneous events and correlation** covering the normalization and correlation of heterogeneous security events.
- **Risk analysis** assessing the level of risk and measuring the impact of potential threats, whether they appear.
- **Decision making support** helping to mitigate the risks and take efficient actions in accordance with the security policy.
- **Digital forensics for evidence** facilitating the reconstruction of malevolent events or helping to anticipate unauthorized actions.
- **Visualization and reporting** providing a dynamic, intuitive and role-based User System Interface for the various stakeholders to use in order to understand the current security situation, to make decisions, and to take appropriate actions.

4.6.4 Technology Choice

The FIspace platform is meant to keep significant amount of personal and business information throughout the system. Personal information might include simple user information as well as national identification number. And business information can be anything such as company’s financial information, partners’ information and other business sensitive data. Any leak or unauthorized access to any of this information can cause catastrophic results. Therefore, priority is to make sure all information is protected and only visible to its rightful owner. The top criteria used in making the technological choices is security.

3.6.1.1 Authentication and Single Sign-On

Authentication is the process of verifying that "you are who you say you are" and it is the backbone of user and software security. Users’ identities should be kept private, but an information owner should be able to access and alter personal data. Therefore, a user should be authenticated prior to making any changes. Users are also categorized based on the authorization levels, and authorization levels allow users to access certain information, certain application and even certain parts of a specific application. To gather this authorization information, first, a user should be authenticated to the system. There are multiple authentication methods to certify transferred data is secure and not altered such as SAML, OAuth, OpenID. Although SAML is used for the prototypes, with the recent advancement on these technologies it is essential to study these technologies once more prior to FIspace system coding. Also, depending on the technologies supported by the GEs, these decisions can be altered to use the preferred GE.

FIspace is being designed to provide two-factor authentication. Two-factor authentication ensures that accepted user credentials are not stolen and not retrieved from the actual owner by force. So two-factor authentication requests information from the user; something only the rightful owner of the credentials might access. Once the user enters the credentials and these credentials are accepted, an SMS message is sent to user’s cellphone which is registered to FIspace system. Then the user is asked to enter the content of the SMS message into an authentication form. As of today, OneIdM pledges the delivery of two-factor authentication in the July release.

Single sign-on (SSO) is an authentication process that permits a user to enter one username and password in order to access multiple related, but independent applications. The process authenticates the user for all the applications they have been given rights to and eliminates further prompts when they switch applications during a particular session.

3.6.1.2 Identity Management and RBAC (Authorization)

Identity management (IdM) is an administration of individual identities within a system, such as a company, a network or even a country. In enterprise IT, identity management is about establishing and managing the roles and access privileges of individual network users. ID management systems provide IT managers with tools and technologies for controlling user access to critical information within an organization.

Role-based access control (RBAC) is an approach to restricting system access to authorized users. It is used by the majority of enterprises and can implement mandatory access control or discretionary access control. RBAC is sometimes referred to as role-based security or simply as Authorization.

Unfortunately, as of today, there is no GE that provides IdM and RBAC. Since identities and roles are the backbone of the entire FIspace system security, the team decided on using LDAP for IdM and security policies of LDAP’s group attributes to deliver a stable security infrastructure.

4.6.5 Release Plan

The initial release plan for the above mentioned features can be found in the table below. This release plan is takes the approximate dates of GEs deliveries into account, and therefore might be changed throughout the duration of the project.

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
OneIdM	Defining requirements for Integration of OneIdM SSO structure with apps.	Refined		Final
OneIdM Integration	Integrating SSO structure to Front-End	Initial	Refined	Final
Identity Management	Defining rules and creating infrastructure (FIspace DB) for Identity Management feature.	Initial	Refined	Final
Authorization	Defining sample rules and building infrastructure (FIspace DB) for RBAC like system.	Initial	Refined	Final
User provisioning	Creation, maintenance and deactivation of user objects and user attributes between two databases	Initial	Refined	Final
Privacy GE Integration and Implementation	Integration of Security. Privacy GE and implementation to access control management system.	Initial	Refined	Final
Data Handling GE Integration and Implementation	Integration of Data Handling GE and its implementation to access control management system. Complete testing will be done by M18		Initial	Final
Access Control GE Integration and Implementation	Integration of Access Control GE and its implementation to access control management system. Complete testing will be done by M18		Initial	Final
Content Security GE	Defining requirements for Integration of Content Security GE with apps		İnitial	Final
Content Security GE Integration or/and implementation	Implementing and integrating the GE. This GE is using Access Control GE internally, so integration to Access Control GE is needed. Testing will be part of this.		İnitial	Final
Malware Detection GE	Defining requirements for Integration of Malware Detection GE with apps		İnitial	Final
Malware Detection GE	Implementing and integrating the GE. Testing will be part of this.		İnitial	Final

Integration or/and implementation				
Security Monitoring GE	Implementing and Integrating Security Monitoring GE with other SPT GEs and Service Provider Firewall / IPS	İnitial	Refined	Final

Table 27: Initial Release Plan FIspace Security-Privacy-Trust

4.7 Development Environment

The FIspace platform provides a set of features to be consumed by both apps running over it and/or external systems via its exposed API. In order to develop correct apps a developer must understand the behaviour of the platform and how its functionality is exposed.

An app must be understood as a widget, or a mash-up of them, which can be executed on the FIspace web-site, both on a computer or optimized for a mobile device. Technically speaking a widget will be developed using HTML5 [14], JavaScript[15] and CSS[16].

4.7.1 SDK for App Developers

4.7.1.1 Overview

The Development Environment is in charge of providing to the app developers a SDK (Software Development Kit) to ease their work during the implementation of the apps, providing some specific tools and hiding the complexity of the platform. Its main functionalities are:

- Ease the development of the UI of the app, first providing some guidelines, later some templates according to the FIspace specification and finally, if it is feasible, some visual editor or wizard.
- Ease the connection of the backend of the app with external systems
- Definition of Business Collaboration Objects (BCO) and Events to run the B2B core of the platform
- Upload the app into the Store of the platform, including the Linked USDL definition

All of them compliant with the security features embedded within the platform.

4.7.1.2 Main Features

Due to this list of different features the work to be done within the T280 is closely related to the other tasks of the WP200.

The IDE chosen to develop the SDK must be easily extensible, among other characteristics, as further explained in the section 4.7.4. Therefore, the SDK will be developed under the shape of a plugin, for the chosen IDE, composed by a bunch of atomic plugins. The SDK will provide the following features, as reflected in Figure 32:

- Creation of a new “FIspace App project” wizard, choosing the features to add, and based on that creating a specific folder structure.
- Provide a tool to define the UI of the app, possibly using a visual editor.
- Provide a tool to define the integration with communication with external services, possibly using an editor. As further explained in the section 4.4, the app developer shall choose between one of the available interfaces to connect with legacy/external systems, provided by the platform, and to implement it choosing one message exchange protocol (REST, CORBA, JMS, etc.). Afterwards, when the app is instantiated by the final user, this will have to choose one of the provided implementation and to provide the contact details of its own external system to complete the connection of the platform with it.
- Provide a tool to define BCO, possibly using a visual editor as the one provided by the ArtiFact system[17].

[14] http://www.w3.org/html/wg/drafts/html/master/
[15] http://www.w3.org/standards/techs/js#w3c_all
[16] http://www.w3.org/Style/CSS/Overview.en.html
[17] https://sourceforge.net/projects/bizartifact/

- Upload of the app into the platform Store, checking the correct composition of the app and that it is compliant with the security of the platform, possibly using a wizard.

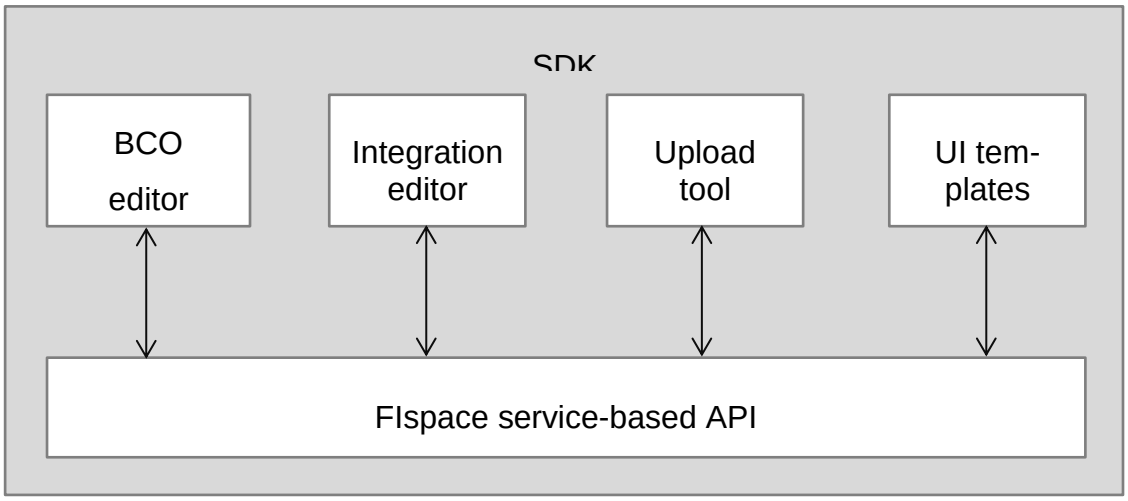


Figure 32: Illustration of SDK Plugins and API

The API is a collection of libraries providing several of the functionality exposed by the platform, which will be used by the different tools previously described. Together to these tools it composes the SDK.

Each one of the previous tools will define a “descriptor” related to the app, which together to the defined UI and other necessary documentation, as a USDL description of the tool, will be uploaded into the platform.

4.7.2 Generic Enablers and Technology Choice

4.7.2.1 Generic Enablers

None of the GEs provided by FI-WARE will be used directly while developing the SDK since functionalities provided by the GEs are not suitable to develop any component of the SDK provided by the development environment. GEs will be used indirectly via FIspace WP200 modules including System & Data Integration, B2B Collaboration core, FIspace Store and Enterprise Service Bus.

4.7.2.2 Initial Technology Choice

The FIspace Development Environment has to deal with different development, deployment and executions constrains from the technical point of view, but also has to cover the FIspace service model. In this context we analyze different technology alternatives for supporting the SDK.

Concerning to the **development environment** the selection requirements are:

- Open technology platform
- Extendible and modular platform
- Support for the execution environment languages
- Multi OS.

Regarding to the **execution environment,** the selection requirements should be classified into two groups: technical requirements and business and service model requirements.

Business and service model Requirements:

- Support for Cloud
- App/Service Market

- Easy integration with FI concepts, for example a GE, and legacy systems such as social networks, web resources and so on.

Technical Requirements:

- Multi OS support
- Multi-Platform support
- Working with any plugins or auxiliary software.
- Access to Mobile Capabilities
- Efficient execution environment. Easy software portability.

After a review of literature, we present the following candidates:

- SDK Core system
  - Eclipse
  - Netbeans
  - Komodo IDE
  - Gedit
  - Microsoft Visual Studio
- Languages for developing functionalities and software
  - Java
  - C++
  - C#
  - HTML
  - Perl
  - HTML5/JavaScript

The selected elements for building up the SDK complies with all the requirements previous exposed. The selection of Eclipse as the central element of the SDL, also enables the SDK to be open source and fully adaptable to changes, not only in the GUI part but also in the underlying parts. Eclipse has been designed with modularity in mind, so it offers a suitable environment for developing the kind of applications and functionalities needed inside the project.

Other alternatives such as Netbeans and Komodo are also valid but it is certainly true that they do not offer the same level of dynamism and community support than Eclipse. The latter is compatible with many of existing programming languages (of our context) with pluggable components; so, it is another advantage that we have taken into consideration. Eclipse also supports cloud-based development through the Orion framework.

We choose HTML5/JavaScript technologies, for the execution environment, as they are proven to be ubiquitous and portable in almost any existing device. In addition, even if some browsers do not yet fully comply with the standard, there are strong efforts for adapting them for this purpose.

HTML5/JavaScript technologies are independent from the device as browsers act as the execution platform, which are likely to be reliable for the API, as functionalities they aim to support. Furthermore, future operating systems (especially mobile ones) are based on HTML5/JavaScript technologies, so it promotes the code reusability, instead of portability, as code can be updated on-the-fly.

Finally HTML5/JavaScript technologies are fully supported by the Eclipse SDK, which allows a straightforward development of applications, expected SDK modules, or libraries.

4.7.3 Development environment Conceptual Architecture

In this section, an overview of the Conceptual Architecture of the SDK is presented. The architecture consists of several components, shown in the Figure 33:

- Hosting IDE: The main block which is the IDE for the development of the app

- Platform parts with which the Hosting IDE component interacts:
  - The FIspace Store
  - The B2B Collaboration Core
  - The Enterprise Service Bus (ESB) which realises all the communication between the different FIspace components
- A component representing the External/Legacy Systems to be integrated within FIspace
  External services to be also integrated in the SDK, e.g. Linked-USDL Editor.

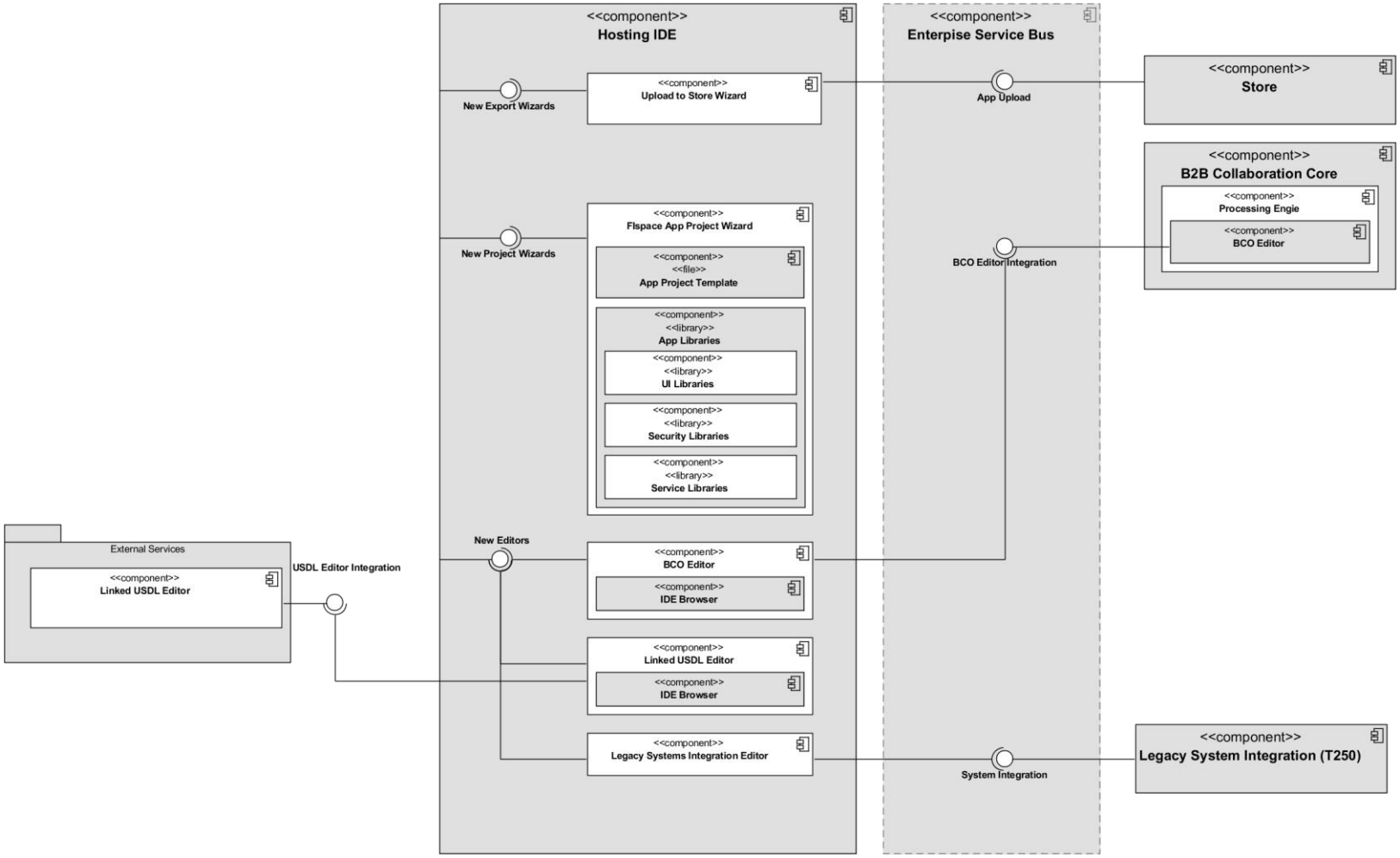


Figure 33: Conceptual architecture of the SDK

4.7.3.1 Hosting IDE Environment

This is the main block of the overall High-Level Technical Architecture. Most probably Eclipse is going to be used, being a suitable environment and providing the necessary modularity required for the task’s needs.

The IDE component will consist of three main sub-entities:

- The Export Wizard: The tool, which will provide the “Upload to FIspace Store” functionality. The Wizard will use the Store API (service-based).
- The Project Wizard: The tool, which will create a new project for developing a new FIspace App. This includes:
  - The App Project Template: The necessary structure of folders and files of the project (e.g. HTML-specific files folder, JS-specific files folder, Java-specific files etc.)
  - Necessary App Libraries: Libraries to be used for the development of the App; mainly FIspace Core modules libraries like the UI library, the Security Library etc.
- The Editors which will be part of the SDK, either as internal or external components:
  - BCO Editor: Will use the internal browser of the IDE to connect to the BCO Editor of ACSI used in T240. An external service, the interaction with the SDK will be realized via the ESB.
  - Linked-USDL Editor: Similarly with the BCO Editor, it will be used via the internal IDE Browser, as an external Service. Again the ESB will be the means to realize the interaction between these components.
  - Legacy System Integration Editor: A particular editor has to be developed which will enable the procedure of integrating specific legacy systems into the Business Processes. This editor should in fact enable (i) either the process of the App instantiation (by the Business IT Engineer), after it has been downloaded by a user from the FIspace market, or (ii) the integration of a specific Legacy System during the development of the App (by the App Developer this time). The Legacy System Integration Editor will configure the Legacy Systems in order to import/export data and interact with the FIspace Apps.

4.7.3.2 The FIspace Store

All the applications, which are going to be developed within the FIspace context using the provided SDK will then be uploaded to the FIspace Store. An appropriate tool has to be developed as a result, which will be used by the app developer to upload the new Fispace App to the store. The interaction between the FIspace Store component and the Hosting IDE component will be realised via the Enterprise Service Bus, as for any Fispace components communicating with each other.

4.7.3.3 The B2B Collaboration Core

The B2B Collaboration will be one of the FIspace Core Modules which will be used by many other components and Apps. Its purpose is to manage, execute and monitor collaborative processes at the heart of FIspace. As already mentioned above, in the Editors part, the internal IDE Browser will be used to connect to the external BCO Editor of ASCI, and as a result to the B2B module, via the ESB module.

4.7.3.4 The Enterprise Service Bus

Already mentioned several times in the description of the above components, the ESB is an essential part of the overall architecture, bringing together the various components, internal or external to the FIspace platform.

4.7.3.5 The Legacy Systems Integration Component

This component will be actually the toolkit provided by the T250 (System & Data Integration), which will be bundled by the SDK.

4.7.3.6 External Services Component

The Linked-USDL Editor is one of the External Services to be integrated and used by the FIspace SDK. As presented in the architecture, and also mentioned above in the Editors' section, via the internal IDE browser this external service will be used by the SDK.

4.7.4 Release Plan

Table 28: Release Plan for the FIspace Development Environment

Feature	Description	V1 (M9/12)	V2 (M15/18)	V3 (M21/24)
New Export Wizard	The tool, which will provide the "Upload to FIspace Store" functionality. The Wizard will use the Store API (service-based).		Initial	Final
New Project Wizard – App Project Template	The necessary structure of folders and files of the project (e.g. HTML-specific files folder, JS-specific files folder, Java-specific files etc.)		Initial	Final
New Project Wizard – App Libraries Component	Libraries to be used for the development of the App; mainly FIspace Core modules libraries like the UI library, the Security Library etc.		Initial	Final
New Editors – BCO Editor	Will use the internal browser of the IDE to connect to the BCO Editor of ACSI used in T240. An external service, the interaction with the SDK will be realized via the ESB.		Initial	Final
New Editors – Linked USDL Editor	Similarly with the BCO Editor, it will be used via the internal IDE Browser, as an external Service. Again the ESB will be the means to realize the interaction between these components.		Initial	Final
Legacy Systems Integration Editor	A particular editor has to be developed which will enable the procedure of integrating specific legacy systems into the Business Processes.		Initial	Final
User Registration	Create a user profile (new user), and associate to company profile (existing)	Initial	Refined	Final
Selection of Technology candidates	The initial list of selected technologies for creating the SDK	Initial	Refined	Final
Selection of Supporting Framework for IDE creation	Among all the studied framework the most suitable is going to be selected for supporting the IDE	Initial	Refined	Final
Components and configuration of components	Selected technologies, represented by specific instances shall be configured	Initial	Refined	Final
Planned GEs to be used & validated	GES provided by FI-WARE are analysed in order to selected the suitable for being integrated into the SDK.	Initial	Refined	Final

5 Conclusions and Outlook

As the first Deliverable of WP200 that is concerned with the development of the generic software infrastructure for the FIspace to enable envisioned seamless collaboration in business networks and support the development, provisioning, and consumption of FIspace Apps, this report has presented the initial conceptual and technical design of the FIspace, including a the overall concept and initial high-level architecture, the agile development methodology applied in WP200 and throughout the project as a whole, and the initial consolidated technical design of the main FIspace components that are development in WP200 along with the initial validation of the Generic Enablers from the FIWARE project that are planned to be used for the implementation and the initial release plans for the planned features. With this, the present report provides the initial results of tasks T210 – T280 as defined in the Description of Work (DoW); it is to note that, following the agile methodology, the initial technical design and work plans presented here will be refined and revised where necessary throughout the upcoming project milestones.

Summarizing the main results, Section 2 has presented the overall design of the FIspace, including a comprehensive overview of the overall concept depicting the business relevance and introducing the main technical building blocks, an explanation of the overall operational model at hand of an illustrative example referring to the Greenhouse Trail (cf. subtask T422 in WP400) which has served as one of the sample use cases for the scenario-driven technical design, and the identification of the main usage processes as well as the workflows of the principal user groups for working with the FIspace.

Then, Section 3 has defined the development plan and methodology for WP200, in particular the overall structure and focus of the release plan for the three main releases planned throughout the project (V1 in M9, V2 in M15, and V3 in M21 with respective maintenance updates), and introduces the agile methodology that has been defined and established throughout the project by adapting modern professional software development techniques to the specific set-up and needs; this is particularly applied in WP200 for the FIspace development as well as within the related work packages in the other work packages in order to ensure proper alignment and allow for agile adaptation of activities with respect to specific demands, requirements, and feedback that is expected to arise throughout the project duration.

Based on this, Section 4 has provided the detailed initial technical design of the seven main components of the FIspace that are developed in WP200. This has been elaborated in an iterative manner based on scenarios and already known requirements from the FIspace trials with main attention to a coherent specification of the planned features and interoperability of the components, so that the presented technical design represents the initial version of the consolidated conceptual design of the FIspace. For each of the components, the main features are defined, an initial assessment of the relevant Generic Enablers from FIWARE is provided along with additional technology choices considered for the implementation, and an initial release plan of the specific features is defined. Summarizing, the main FIspace components are:

(1) The *‘FIspace Front-End’* as the main point of access to all FIspace features and apps, which consists of the following main features and functionalities:
- The Core Front-End that provides the main access page for users,
- The integrated access to FIspace Apps for individual users,
- Personalization and configuration support to adopt the FIspace to individual user needs,
- Build-in social networking and business collaboration features for enabling the seamless communication with business partners and manage individual business networks, and
- Ubiquitous access to allow using the FIspace anywhere via any device;

(2) The *‘FIspace Store’* that provides the marketplace for FIspace Apps, including support for
- Provisioning, i.e. the infrastructure for supporting App Developers to offer FIspace Apps
- Discovery, i.e. the infrastructure for finding and investigating available FIspace Apps, and
- Purchasing, i.e. the tool support buying process for FIspace Apps;

(3) The *‘Real-time B2B Collaboration Core*’ modules that enable the event-driven provisioning of information on collaborative business activities to all involved actors at the right time, consisting of:
- The Business Collaboration Module (BCM) that supports the modeling, execution, and flexible adaptation of collaborative business processes, and
- The Event Processing Module (EPM) that monitors events from several sources and detects situations of interest, therewith making the FIspace an event-driven platform;

(4) The *'System & Data Integration'* facilities as the tool-supported infrastructure to allow the user-controlled connection and of existing system landscapes and external systems to the FIspace, incl.:
- Support for integration of Business & Legacy Systems already used by FIspace users,
- For connecting Internet-of-Things (IoT) enabled systems such as e.g. sensor networks,
- Tool support for handling and integration of heterogeneous data;

(5) The *'Operating Environment'* that ensures the technical interoperability of the FIspace components and its reliable operation, including the following main features:
- The Cloud Service Bus (CSB) as the main middleware for actual data interchange among all FIspace components and FIspace Apps, along with the necessary monitoring facilities,
- Consistency Services that ensure the reliable and fault-tolerant operation of the FIspace;

(6) The *'Security, Privacy, and Trust'* (SPT) framework that ensures the FIspace to be secure by design by providing the necessary security technologies and mechanisms to ensure access control, information confidentiality, and trustworthiness, including in particular:
- Sophisticated access control infrastructure ensuring that all information on the FIspace can only be access with valid authorization (identity management and authentication),
- Ease the access to authorized users (e.g. by Single-Sign-On),
- Ensure secure execution and attack handling of the FIspace as a cloud platform,
- A through tool-supported infrastructure to ensure that the mandatory security guidelines are applied by all FIspace component and App developers;

(7) the *'Development Environment'* that provides a software development kit (SDK) to support the quick and technically valid development of FIspace Apps and their instantiation for End-Users, including:
- The basic SDK-environment for App development
- Integrated tool support for using FIspace technologies for the App development, in particular re-usable UI-libraries, infrastructure for using and working with the BCM and EPM modules, and the integration of external systems (back-end, legacy, IoT)
- Assurance of the technical governance, i.e. that every App properly applies the mandatory security mechanisms and is running in the FIspace Operating Environment.

With this, this report has provided a comprehensive and consolidated initial conceptual design and technical architecture for the generic FIspace software infrastructure along with the initial release plans in alignment with the other work packages has been elaborated, so that WP200 has achieved the goals planned for milestone M3 of the project.

For the next milestone, it is planned to build upon this and refine the overall technical design towards the coherent technical specification of the FIspace and prepare for the implementation for the first main release that is planned for M9 of the project. In particular, the goals for the M6 project milestone are:

- Elaborate the technical specification for the FIspace components and their overall interaction
- Define the procedures for usage and validation of the Generic Enablers and establish the contacts for direct interaction with the respective GE providers from the FIWARE project, in accordance to the overall GE validation strategy as part of the overall alignment of the FIspace project with the FI PPP program that is defined in WP100
- Establish the procedures and development tool support for the deployment on the FIspace hosting infrastructure in close alignment with WP300
- Continue the direct interaction with the trial teams from WP400 as well as the FIspace App development in T450, as an essential part of the user-driven design with immediate and iterative feedback and validation of the FIspace
- Close collaboration and alignment with the elaboration of the overall business model and ecosystem incubation activities driven by WP500 in order to ensure high quality and early engagement of external stakeholders and the preparation for FI PPP Phase 3.

References

[1] H. Kisker, “The Changing Cloud Agenda: cloud computing shifts from cost to innovation,” 2012.

[2] FIWARE, “FI-WARE Product Vision,” *2012*. [Online]. Available: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Product_Vision.

[3] T. Dingsøyr, T. Dybå, and N. B. Moe, *Agile Software Development: Current Research and Future Directions*, 1st Editio. Springer, 2010.

[4] H. Plattner and C. Meinel, *Design Thinking Research: Studying Co-Creation in Practice (Understanding Innovation)*, Edition 20. Springer Berlin / Heidelberg, 2012.

[5] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley, 2009.

[6] H. Richard, E. Damaggio, F. Fournier, M. Gupta, F. T. Heath, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculin, “Introducing the Guard-Stage-Milestone Approach for specifying Business Entity Lifecycles.,” in *Proceedings of the 7th international conference on Web services and formal methods (WS-FM’10)*, 2010, pp. 1–24.

[7] O. Etzion and P. Niblet, *Event Processing in Action*. Manning, 2010.

[8] Y. Engel, O. Etzion, and Z. Feldman, “A Basic Model for Proactive Event-Driven Computing,” in *Proceedings of the sixth ACM conference on Distributed Event Based Systems (DEBS’12)*, 2012.

[9] Y. Engel and O. Etzion, “Towards Proactive Event-Driven Computing,” in *Proceedings of the fifth ACM conference on Distributed Event Based systems (DEBS’11)*, 2011, pp. 125–136.

[10] A. Metzger, R. Franklin, and Y. Engel, “Predictive Monitoring of Heterogeneous Service-oriented Business Networks: The Transport and Logistics Case,” in *SRII 2012 Global Conference*, 2012.

[11] Z. Feldman, F. Fournier, R. Franklin, and A. Metzger, “A Case Study on the Proactive Management of Transport Processes,” in *Proc. of the 7th ACM International Conference on Distributed Event-Based Systems, June 29 - July 3, Arlington, Texas, USA*, 2013.

[12] A. Metzger, R. Franklin, and Y. Engel, “Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case (best service engineering innovation & quality paper),” in *Proc. of SRII 2012 Global Conference*, 2012.

[13] G. Chockler, I. Keidar, and R. Vitenberg, “Group Communication Specifications: A Comprehensive Study,” *ACM Computing Surveys*, vol. 33, no. 4, pp. 427–469, 2001.

[14] V. Bortnikov, G. Chockler, D. Perelman, A. Roytman, S. Shachor, and I. Shnayderman, “FRAPPE: Fast Replication Platform for Elastic Services,” in *LADIS*, 2012.

[15] R. Melamed and I. Keidar, “Araneola: A Scalable Reliable Multicast System for Dynamic Environments,” *Parallel Distributed Computing*, vol. 68, no. 12, 2008.

[16] S. Girdzijauskas, G. Chockler, Y. Vigfusson, Y. Tock, and R. Melamed, “Magnet: Practical subscriptions clustering for Internet-scale publish/subscribe,” in *DEBS*, 2010.

[17] V. Bortnikov, G. Chockler, A. Roytman, and M. Spreitzer, “Bulletin board: a scalable and robust eventually consistent shared memory over a peer-to-peer overlay,” *SIGOPS Operation System Review*, vol. 44, no. 64–70, 2010.

[18] D. Breitgand and A. Epstein, “SLA-aware placement of multi-virtual machine elastic services in compute clouds,” in *IFIP/IEEE International Symposium on Integrated Network Management*, 2011, pp. 161–168.

[19] E. Dekel and G. Goft, “ITRA: Inter-Tier Relationship Architecture for End-to-end QoS,” *Journal of Supercomputing*, vol. 28, pp. 43–70, 2004.

[20] E. Dekel, “A Peer-to-Peer Middleware Platform for Highly Scalable and Dependable Services,” in *Keynote at INSERTech@autonomics Workshop on INnovative SERvice Technologies, Rome, Italy*, 2007.

[21] E. Dekel, G. Gershinsky, and C. Martin, “Using WAS XD with Reliable and Consistent Message Streaming for High Volume Order Matching,” in *IBM Academy of Technology Conference on High Availability Best Practices*, 2006.


FIspace
Business Collaboration