D6.1

ARCHITECTURE SPECIFICATION

April 2014

ABSTRACT

This document contains the architecture specification of the FIcontent platform. This includes the description of the three domain-specific platforms (i.e. Social Connected TV Platform, Smart City Platform, and Pervasive Games Platform) and a description of the Specific Enablers dedicated to the current state of the platforms. Moreover, the Specific Enablers are explained, which are utilized by multiple platforms and thus promoted to be Common Specific Enablers of FIcontent. Finally, the documentation about the usage of Generic Enablers from FI-WARE is given.

This document is a deliverable of the FI-CONTENT 2 integrated project supported by the European Commission under its FP7 research funding programme, and contributes to the FI-PPP (Future Internet Public Private Partnership) initiative.

DELIVERABLE DETAILS

[Full project title]:	Future media Internet for large-scale CONTent experimENTation 2
[Short project title]:	FI-CONTENT 2
[Contract number]:	603662
[WP n°]:	WP6 – Technology Enablers
[WP leader]:	Mario LOPEZ-RAMOS, Thales
[Deliverable n°]:	D6.1
[Deliverable title]:	Architecture specification
[Deliverable nature]:	Report (R)
[Dissemination level]:	Public (PU)
[Contractual delivery date]:	M12 – March 2014
[Actual delivery date]:	29 April 2014
[Editor]:	Philipp Slusallek, DFKI
[Internal Reviewers]:	Kenny Mitchell, BLRK / Martin Gordon, RBB
[Suggested readers]:	Executives in entertainment companies and banks, investors
[Keywords]:	Platform, Enabler, Social Connected TV, Smart City, Pervasive Games
[File name]:	FI-CONTENT 2_WP6-005_D6.1_V2.0

EXECUTIVE SUMMARY

This document contains the architecture specification of the FIcontent platform. This platform consists of three domain-specific platforms, each providing the technological foundation to enable the creation of respective applications in the domain of Social Connected TV, Smart City Services and Pervasive Games.

The core of the Social Connected TV Platform can be seen as a toolbox offering powerful tools to enhance connected TV services or TV related services for second-screen devices. The Smart City Platform is a portfolio of functions, designed to foster the development and uptake of Smart City Applications based on Future Internet technologies. In addition, the Smart City Platform will enable end-users to generate usecase-specific applications including dedicated features of the platform. Furthermore, the Pervasive Games Platform is a set of integrated, modular Enablers designed to aid building Internet-based games connected with the real world. While moving from native to in-browser execution as browser technology develops, the Pervasive Games Platform targets real-time, low latency, and high performance goals.

Each platform is a collection of established development tools, specific technical contributions (FIcontent Specific Enablers), and generic Future Internet technology (FI-WARE Generic Enablers). The Specific Enablers that are particularly developed for each platform within FIcontent are listed below.

Social Connected TV Platform:

- Audio Fingerprinting SE
- Audio Mining SE
- Content Optimisation SE
- Second Screen Framework SE
- TV Application Layer SE

Smart City Platform:

- Open City Database SE

Pervasive Games Platform:

- Leaderboard SE
- Reality Mixer – Reflection Mapping SE, Camera Artifact Rendering SE
- Augmented Reality – Fast Feature Tracking SE, Marker Tracking SE
- Games with Things – Spatial Matchmaking SE
- Game Synchronization SE

We also have identified a set of Specific Enablers used by multiple platforms within FIcontent and share the technical knowledge between the three platforms. Therefore, these Enablers are reclassified to become common Specific Enabler and may have the potential to become generic Future Internet technology in a later stage. Those common Specific Enablers are in particular:

- Social Network SE
- Content Sharing SE
- Content Enrichment SE

In order to achieve the desired functionality of applications and to realize all of those SEs, we take advantage of generic Future Internet technology by integrating respective useful GEs from FI-WARE into the platforms, such as the 3D-UI GE, which will be utilized within the Pervasive Games Platform and the Smart City Platform to provide an extensive experience of gaming scenarios and in order to supply Augmented Reality functionality on the web. We are currently using seven Generic Enablers from FI-WARE and we plan to test and integrate more than twenty GEs in total.

The presented architecture of the FIcontent platform and the Specific Enablers are subject to change during the course of the project. Thus, the document reflects the current state and planning of the FIcontent

platform. We maintain an up-to-date documentation of the architecture and the SE specification within the FIcontent Wiki available at http://wiki.mediafi.org/. Please refer to the Wiki to gather recent information.

LIST OF AUTHORS

Organisation	Author
DFKI	Stefan Lemme Philipp Slusallek
ETHZ	Marcel Lancelle Fabio Zund
DRZ	Chino Noris Mattia Ryffel
BLRK	Kenny Mitchell
FhG/IAIS	Michael Eble Sebastian Kirch
PIX	Dirk Krause
Orange	Arnaud Brun
TRDF	Leroy Bertrand
IRT	Christoph Ziegler
FhG/FOK	Robert Seeliger Miggi Zwicklbauer
GOBO	James Callin
TCF	Farid Benbadis Mario Lopez-Ramos
RBB	Martin Gordon
BBC	Chris Needham

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AR	Augmented Reality
CG	Computer Graphics
API	Application Programming Interface
SE	Specific Enabler
GE	Generic Enabler
FAQ	Frequently Answered Questions
XML3D	Three Dimensional Extensible Markup Language
POI	Point of Interest
DB	Database
HbbTV	Hybrid Broadcast Broadband TV
HTML	Hyper-Text Markup Language
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
SEO	Search Engine Optimization
NLP	Natural Language Processing
SME	Small to Medium Enterprise
UGC	User-Generated Content
RTS	Real-Time Strategy
TAL	TV Application Layer
VoD	Video on Demand
WebGL	Web Graphics Language
GPU	Graphics Processing Unit
FI	Future Internet
FI-PPP	Future Internet – Public Private Partnership
GPS	Global Positioning System

1 - INTRODUCTION

1.1 - Purpose of this document

The platforms developed within the FIcontent project are collections of tools and techniques, designed to enable the creation of Social Connected TV applications, Smart City Services, and Augmented Reality video games and interactive experiences, on mobile devices and the web. This technical portfolio takes advantage of established development tools, of specific technical contributions (FIcontent Specific Enablers), and generic Future Internet technology (FI-WARE Generic Enablers).

This document describes the overall architecture of the FIcontent platform. Moreover, this document will present the architecture of the three domain-specific platforms, the utilization of Generic Enablers and Specific Enablers by the platforms including the technical documentation of the Specific Enablers, which are already available.

Please be aware about the fact that this document is generated from the FIcontent Wiki [1]. Thus, the document may sometimes still refer to the FIcontent Wiki. All information in this document is also available online. We suggest using the online version [2] for an enhanced reading experience.

1.2 - Improvements of the second version

This is a second version of this document taking into account improvements suggested by the European Commission after the submission of the first version. They have been taken into consideration as follows:

1. Sections 3 and 4 must follow the format of Section 2. Section 4 is copy/paste from D4.2. Sections 6 to 9 should follow the same format as well.

We revised the architecture descriptions of all three platforms with respect to the project's progress and the harmonization of deliverables. The architecture descriptions follow the same structure with only minor adaptions necessary for specific needs of the respective platforms.

Since Dx.2 deliverables (i.e. D2.2, D3.2 and D4.2) are of nature P (prototype), we amend each with an extract from D6.1, the development roadmap of the respective platform and deployment information of the SEs for further explanations.

The specification of SEs might change over time. Thus, the harmonization is an ongoing process reflected in the FIcontent Wiki [1] and in updated versions of D6.1.

2. According to Figure 3 (page 21) the Social Connected TV Platform-based services do not access the GEs directly. Is this indeed the case? Note that in the other 2 platforms this is not the case. Please clarify in the next version.

The figure was updated to clarify the usage of GEs by applications.

3. The architecture specification ignores the fact that GEs and SEs may run either on the FI-Lab infrastructure or on any of the experimentation sites. This point is not explained in any other deliverable. Please add to the next version a section describing the planned interaction between FI-Lab and the FI-Content2 experimentation sites.

We introduced a new section describing the relation between experimentation sites and FI-Lab (see Section 7). Moreover, we submitted an additional report D6.5 to the European Commission to clarify the usage and deployment of GEs and SEs in FIcontent's experiments with regard to the user scenarios from Dx.1 deliverables.

4. The Open City Database is an SE in this deliverable, but not in deliverable D3.2.

The Open City Database SE was added to D3.2.

However, the Smart City Platform is under major revision due to changes in the consortium and the new version of D6.1 contains a draft of the new architecture of the Smart City Platform. Moreover, this will be concreted until the next release of the three platforms and Specific Enablers in June 2014.

5. Deliverable D6.1 is a crucial deliverable, and will be a somewhat living document. The consortium is requested to submit an updated version at month 12.

An updated version of D6.1 will be submitted hereby.

However, since this deliverable D6.1 is a somewhat living document, it will be updated regularly - next together with the upcoming release of the FIcontent platforms and Specific Enablers in June 2014.

1.3 - List of terms and definitions

- **Enabler**: Software module or web service providing well-specified functionalities, accessible and usable by application developers through clearly-described APIs (Application Programming Interfaces).
- **Generic Enabler (GE)**: An enabler realized by the FI-WARE project or its follow-up sustainability project.
- **Specific Enabler (SE)**: An enabler realized by the FI-CONTENT2 project. Specific Enablers may be layered on top of, or otherwise make use of, Generic Enablers. Please refer to the definition of a FIcontent SE (see Section 2.4).
- **Platform**: A comprehensive combination of technology infrastructure and - Generic and Specific - enablers capable to host and to support development of application software.
- **Application** or **Application software**: Software layered on top of one or several platforms for realizing some (presumably) useful tasks for end-users.
- **Scenario**: Description of foreseeable interactions of users with one or several applications.
- **Experiment** or **Experimentation**: Concrete test with actual users of one scenario in one of the experimentation sites in a given time frame.
- **Functional requirement**: Either calculations, technical details, data manipulation, processing or other specific functionality that define what a system is supposed to accomplish.

2 - Overview of the FIcontent Architecture

The FIcontent platform is split up into three domain-specific platforms, each providing the technological foundation enabling the creation of Social Connected TV applications, Smart City Services, and Augmented Reality video games and interactive experiences, on mobile devices and the web. The figure below illustrates the fields of applications for the Social Connected TV Platform, the Smart City Platform, and the Pervasive Games Platform.

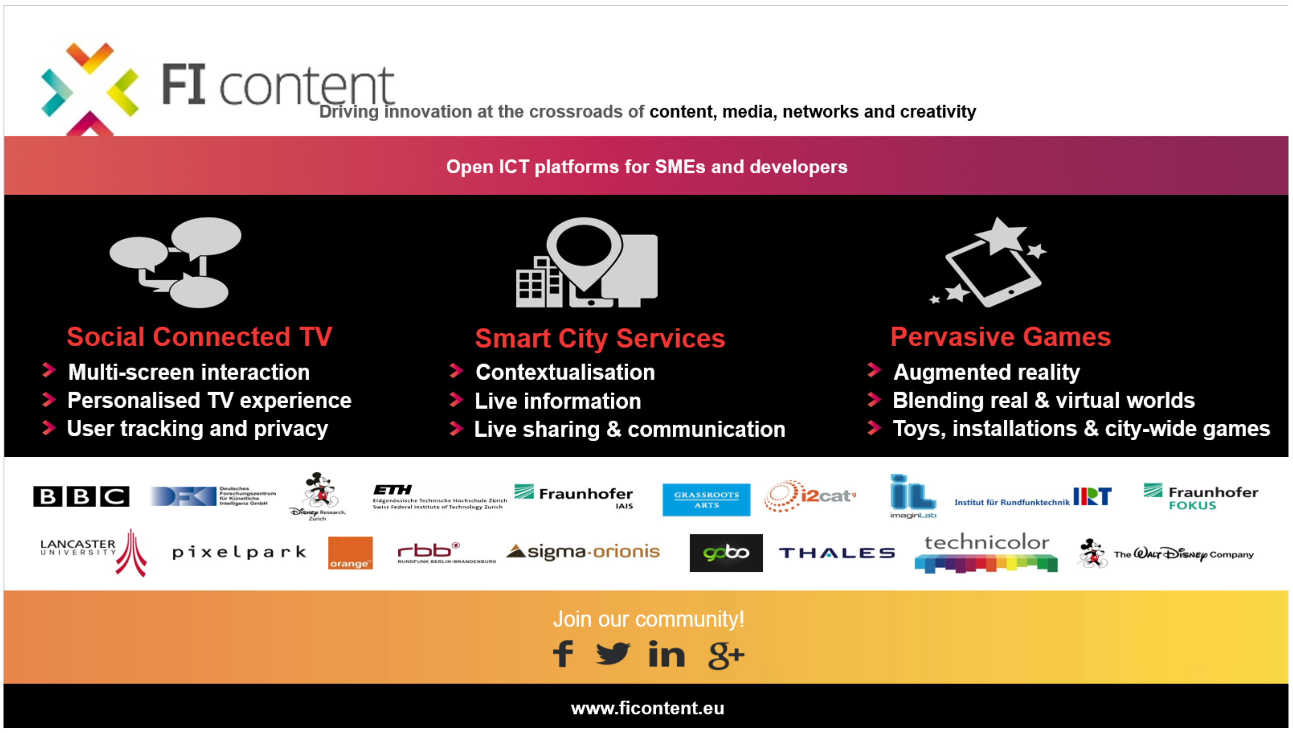


Figure 1 Overview of the three FIcontent platforms: Social Connected TV, Smart City Services and Pervasive Games

Each platform is a collection of established development tools, of specific technical contributions (FIcontent Specific Enablers), and generic Future Internet technology (FI-WARE Generic Enablers). We take advantage of generic Future Internet technology by integrating respective useful GEs from FI-WARE into the platforms, such as the Object Storage GE (see Section 6.2.4), which is utilized within the Social Connected TV and Smart City Platform to store enriched content. Moreover, all web-related technology in the area of the Pervasive Games Platform benefits from the 3D User Interface (3D-UI) GE (see Section 6.2.3) to enable a web-based 3D experience within today's web browser.

Thereby, we approach work within FIcontent collectively and share technical knowledge between the three platforms. This applies to the Identity Management GE (see Section 6.2.1) and Object Storage GE (see Section 6.2.4), for instance. Moreover, we have identified a set of Specific Enablers used by multiple platforms. Thus they were reclassified as Common Specific Enabler (see Section 11) and may later become generic Future Internet technology.

With the presentation of our three platforms we try to create a "platform effect" and attract external developers and SMEs. We make the APIs, documentation, and examples public to simplify the engagement of new developers. Furthermore, we particularly rely on web-based technologies and tools, the ability to share and clone existing software components, and the manageability of the platforms to ease entry into the use of FIcontent technology.

We are using a user-driven and agile development process to adapt the platform's features in order to match gathered feedback and developer needs. In addition, we perform trials of our platforms on various experimentation sites, such as Berlin, Brest, Zurich and Lancaster. Therefore we need to deploy the respective platforms ourselves and thus gain an initial experience to further improve platforms' usability before engaging external developers. The trials will be continued for the Social Connected TV Platform, the Smart City Platform and the Pervasive Games Platform during the whole course of the FIcontent project.

2.1 - Social Connected TV

The Social Connected TV Platform is a toolbox providing powerful tools to enhance connected TV services. This mainly includes Enablers for multi-screen interaction with intuitive interaction for advanced TV services, more versatile content presentation across screens and personalized TV experience. Moreover, the Social Connected TV Platform provides Enablers, which support content portals tailored to single and multiple users with social interaction between users (e.g. explicit recommendation), search and discovery application and user tracking and privacy.Finally we provide Enablers for visualization of personal content consumption by tracking implicit and explicit user interaction, which also provides users with simple control over their personal data. The toolbox design is reflected in the architecture of the Social Connected TV Platform (see Section 3) and allows high flexibility applications can mix and match Enablers to realize their new and innovative functionality.

2.2 - Smart City Services

The Smart City Platform provides an architecture (see Section 4) to enable creative people to generate, share and combine assets, objects and stories to develop mobility city services and new social experiences. Furthermore, it offers contextualization, recommendation, live information, mixed reality, 3D, sharing capabilities, and real-time communication. Moreover, the Smart City Platform contains a social interaction dimension where people can connect and build communities to exchange information via social networks.

The Smart City Platform is accessible for web and application developers and provides dedicated features to easily create Smart City Services with well-known and established technologies. In addition to that, the platform takes advantage of eBiz services, such as the Design My App Portal together with the App Generation module, to support non-developers by allowing them to create use-case-specific applications with their content and according to their needs. Thereby, we introduce a shortcut for creative people realizing their ideas without requiring programming skill.

2.3 - Pervasive Games

The Pervasive Games Platform comprises an architecture (see Section 5) of integrated, modular Enablers designed to aid building Internet-based games connected with the real world. While moving from native to in-browser execution as browser technology develops, we target real-time, low latency, and high performance goals with the Pervasive Games Platform. Moreover, the platform supports new forms of interactive entertainment of three tiers: Digital Consumer Products (Toys), Location-Based Games and City-Wide Games.

The Pervasive Games Platform is accessible for web developers and provides dedicated features to easily create web-based games with well-known and established technologies. In addition to that, the platform takes advantage of the Unity native engine plug-in to support professional game developers by allowing them to work with their accustomed tools.

2.4 - Definition of a FIcontent Specific Enabler

For a common understanding of the term "Specific Enabler" (SE) we provide a selection of properties, which usually applies to all of our FIcontent Specific Enablers.

- **Reusable Software.** A FIcontent Specific Enabler shapes a reusable software package or service to be utilized by FIcontent user scenario applications as well as 3rd-party applications and services.
- **Open Specification.** All Specific Enabler APIs and specifications are publicly available to interested 3rd parties. Thus, we ensure a stable and well-defined API of SEs and the platform effect of the FIcontent platforms.

- **Terms and Conditions.** Each provider of a Specific Enabler supplies together with the SE clear terms and conditions for the usage of the SE by FI-PPP partners, Phase-III participants, and beyond the end of FIcontent project.
- **Accessibility.** Specific Enablers are meant to be reusable by FI-PPP Phase-III projects. Hence, the nature of the SE's terms and conditions should allow usage by 3rd-party developers and SMEs.
- **Interoperability.** FIcontent Specific Enablers can be combined to deliver new cross-domain applications. They are compatible with other Future Internet technology such as Generic Enablers from FI-WARE.
- **Simplification.** Specific Enablers usually simplify either the process of building, deploying or monitoring and managing content-related applications. Thereby, SEs take advantage of cloud infrastructures and Content Delivery Networks so that the appropriate Quality of Service is met.

3 - Social Connected TV Platform Architecture

The figure below illustrates the Social Connected TV Platform architecture. In the figure four layers can be identified. These are:

- Layer 1 (red): Scenarios
- Layer 2 (yellow): Applications
- Layer 3 (blue): Specific Enablers (SEs)
- Layer 4 (green): Generic Enablers (GEs)

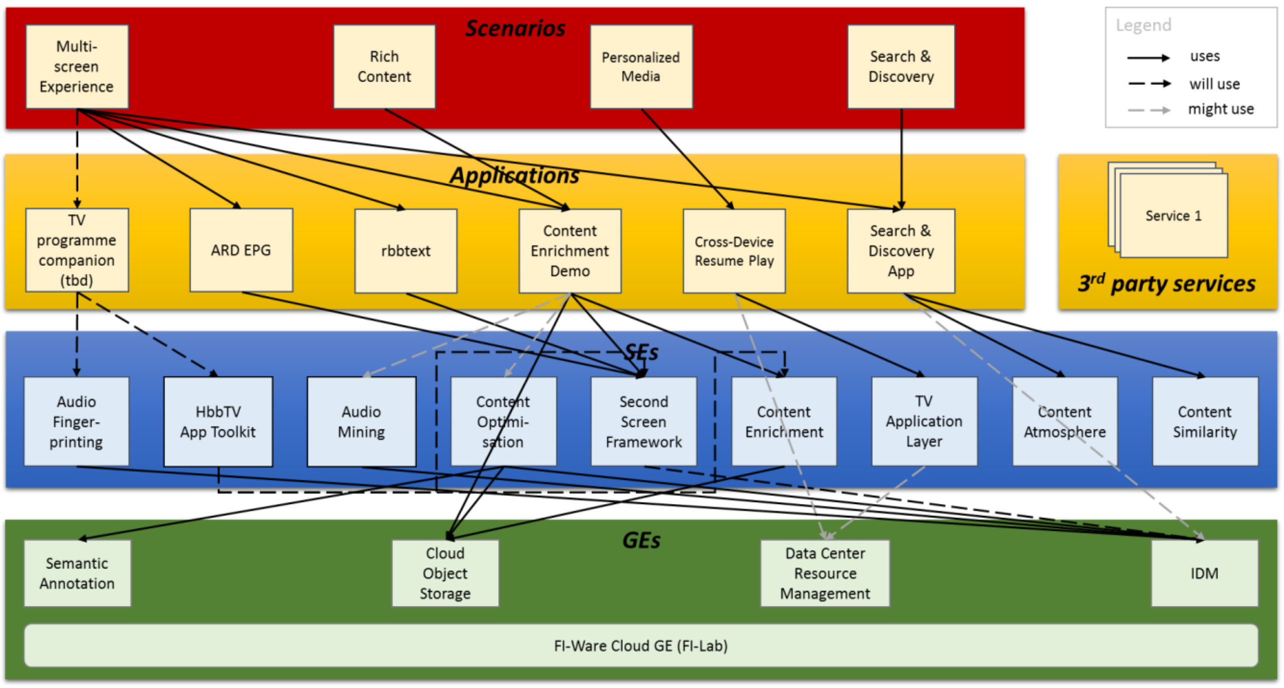


Figure 2 High-level architecture of the Social Connected TV Platform

Scenarios describe what users will be able to do with the Social Connected TV Platform. Scenarios are defined in D2.1. SEs and GEs are technological components that are used to implement the functional requirements of the scenarios. GEs are provided by the Core Platform of FI-PPP developed by the FI-WARE project. SEs are developed by FIcontent in order to complement GEs where domain specific features are needed. SEs are exposed to 3rd party developers and SMEs via the Social Connected TV Platform API.

Applications are pieces of software built on top of SEs and GEs that make certain features of Scenarios available to end-users. There are two kinds of applications. The first group is built of those applications that are implemented by FIcontent partners with the goal of validating the scenarios and the SEs and GEs. The second group is built of those applications that are developed by third parties that are involved through the Open Call, Competitions and Phase 3 of the FI-PPP programme.

The arrows in the diagram indicate dependencies between scenarios, applications and Enablers. A solid black arrow indicates that a certain link is already implemented, e.g. the ARD EPG is actually used for testing the Multi-Screen Experience scenario and the application integrates the Second-Screen Framework SE. The dashed black arrow indicates a planned usage, e.g. the Second-Screen Framework will use the IDM GE in order to allow application developers creating a personalised second-screen experience, but the GE is not yet (fully) integrated. A dashed grey arrow indicates that the use of a component is considered, e.g. it is considered to use the IDM GE to implement authentication mechanisms in the Search & Discovery App.

There is one GE in the diagram that has no connection with an arrow, i.e. the FI-Ware Cloud GE (FI-Lab). This GE is used by almost all SEs and a number of applications for web hosting. We did not explicitly indicate the dependencies for this enabler to prevent the picture from ending up in dense mesh of arrows. A detailed overview on the use of the FI-Ware Cloud GE (FI-Lab) is given in D6.5.

3.1 - Architecture Description

Experiments conducted on the basis of the Social Connected TV Platform aim to gain findings on four major topics. These are:

- Rich Content
- Multi-screen Experience
- Search & Discovery
- Personalised Media

This section introduces how the components of the Social Connected TV Platform are used to augment TV services with features of the above mentioned domains. Moreover we provide a more detailed description of the Specific Enablers [3].

3.1.1 - Rich Content

The goal of the Rich Content scenario is to gain findings on the consumption, annotation and sharing of interactive TV content and additional available media related to it.

Media/Text Annotation (Content Augmentation) is handled by the Content Enrichment SE (see Section 11.3). The implementation covers the HbbTV-enabled TV set as well as the second-screen application which allows the user to edit and receive content related supplemental information. We are currently focussing on the TV implementation to allow the display and utilization of enriched content via the HbbTV application based on the Content Enrichment metadata model. This includes the time synchronized display of additional information for objects in the video scene and the user interaction with them. Functions to edit and receive content on the companion device will be implemented afterwards to extend the interactive content experience across multiple devices.

Object database and storage functions for additional information as video, text, images, are handled by the Object Storage GE (see Section 6.2.4) as provided by FI-WARE. The GE is installed on a XIFI node based in Berlin.

The Audio Mining SE (see Section 8.2) and the Content Optimisation SE (see Section 8.3) can be used to pre-process audio-visual data and generate corresponding metadata automatically. The Audio Mining SE detects speech segments in a video and then transforms the spoken words of these segments into text (speech-to-text). The generated text can then be used to either search within the video, or to link keywords with additional content using the Content Optimisation SE. This Enabler allows users to perform a Named Entity Recognition (NER) on the generated text transcripts and to store the linking information in an object database.

3.1.2 - Multi-Screen Experience

The Social Connected TV Platform eases the creation of TV-programme-related companion applications on connected mobile devices like tablet PCs or smartphones by two complementary technical means, i.e. the Second-Screen Framework SE (see Section 8.4) and the Audio Fingerprinting SE (see Section 8.1).

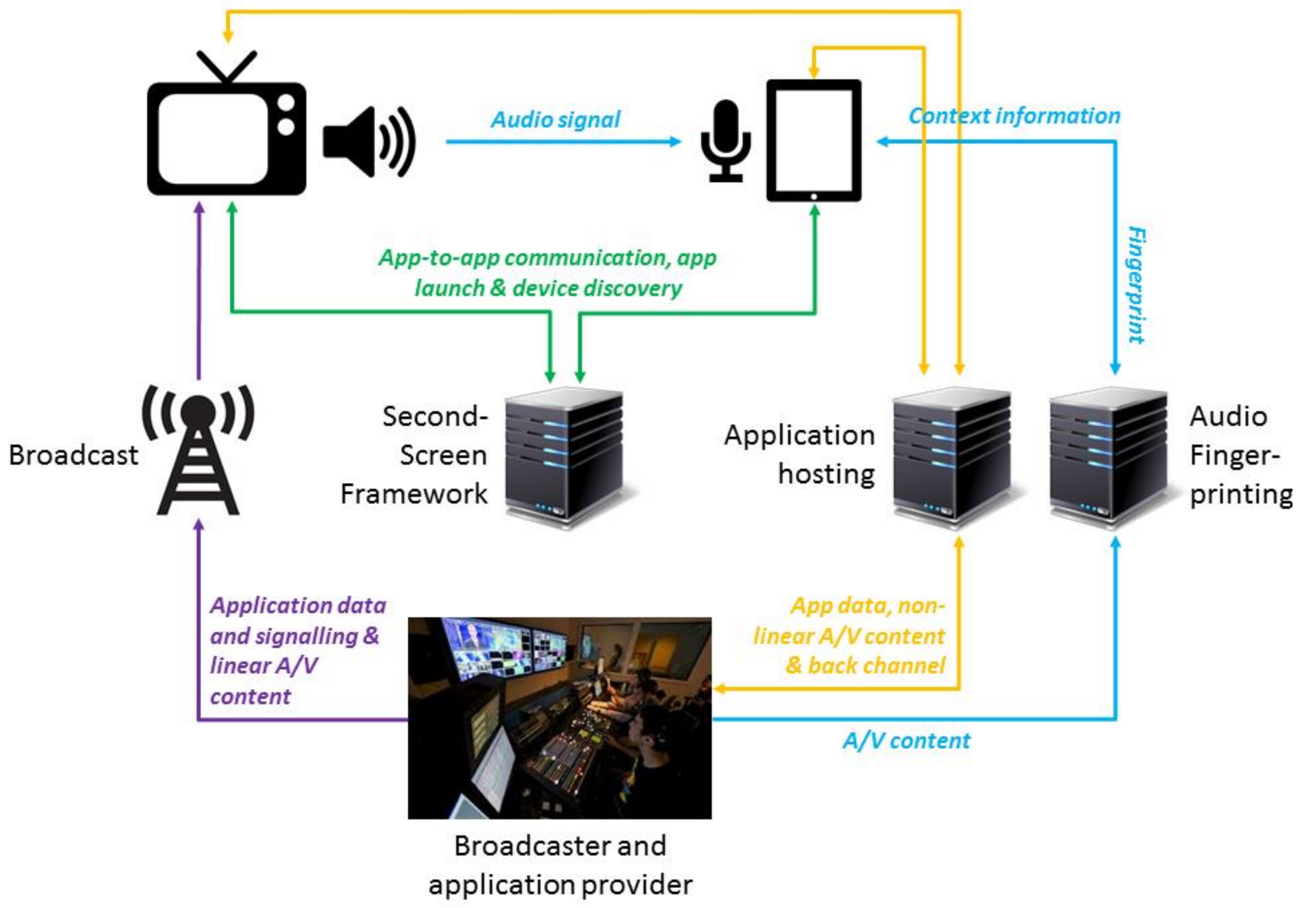

Audio signal
Context information
App-to-app communication, app launch & device discovery
Fingerprint
Broadcast
Second-Screen Framework
Application hosting
Audio Finger-printing
Application data and signalling & linear A/V content
App data, non-linear A/V content & back channel
A/V content
Broadcaster and application provider

Figure 3 Realising applications that allow Multi-screen Experience with the Second-Screen Framework SE and the Audio Fingerprinting SE

The Second-Screen Framework allows creating web apps for mobile devices that are able to exchange information with a web application on a connected TV. Connected TVs that implement the HbbTV standard can interpret application data that is sent with the broadcast signal. This allows broadcasters to provide connected TV applications that go with the programmed content. The bi-directional communication path and the mechanism for automatic application launch allow keeping the second screen in contextual sync with the hybrid TV application.

The Audio Fingerprinting SE facilitates the creation of second-screen applications that go with the programmed content event if the end-users do not have a connected TV. Context awareness is achieved through analysis of the audio signal of the content played back on the TV. Although the second screen cannot be used to interact with the content on the TV, still users avail themselves of the mobile device as display for contextually synced additional information.

The above described set-up allows broadcasters offering interactive programme accompanying content on second screens to a broad audience. It enables them creating a comprehensive Multi-screen Experience.

3.1.3 - Search & Discovery

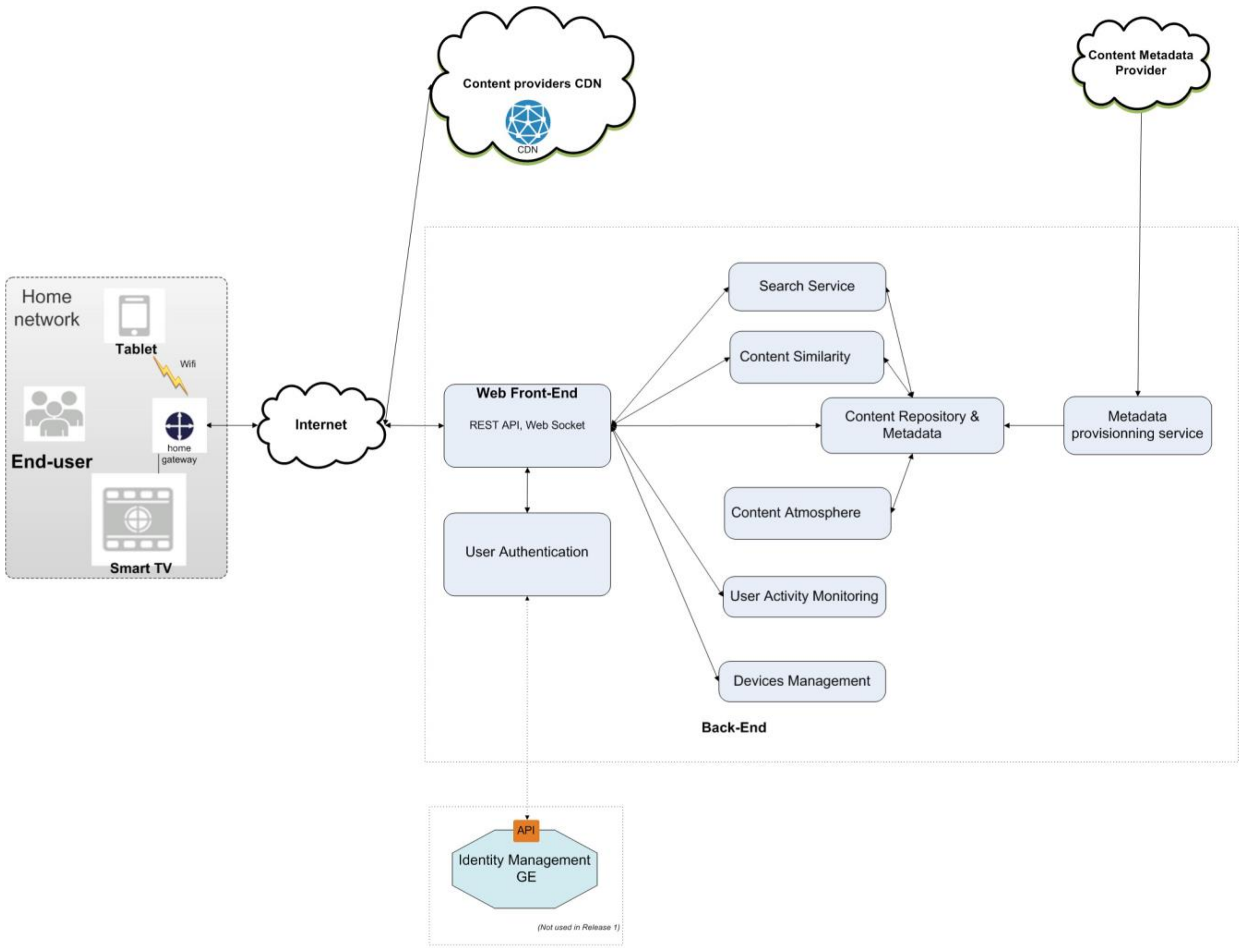


Figure 4 Realisation of applications in the area of Search & Discovery

Today, searching for interesting content to watch is time-consuming for the majority of commercial VoD offers. This scenario intends to explore several ways to explore, search and discover new interesting content for users. This scenario includes the following functionalities running on a tablet:

- An advanced search, with auto-completion, handled by “search service” component.
- A discovery function based on similarity, handled by the “Content Similarity” component. Starting from a movie a user likes, he is able to navigate inside a graph to discover other movies with common aspects, such as same actors, same director, or similar movies proposed by a content to content recommendation engine.
- A discovery function enabling movie selection by indicating multiple criteria such as genres, people, countries, production years, handled by “Content Repository and metadata” component.
- A discovery function based on the combination of genres, handled by both “Content Similarity” component and “Content Repository & metadata” component. A predefined list of cocktails are proposed to the user. User is also invited to create his/her own cocktail by combining its favourite genres.
- A discovery function based on screenshot, handled by “Content Repository and metadata” component: five lists of screenshots are proposed to the user without any additional information. Each list proposes screenshot of a single movie. When the user selects a list he/she gets the movie detail page and discovers which movie was behind these screenshots.

Discovery based on the mood of movies is handled by the "Content Atmosphere" component. This feature has not been evaluated during this first experimentation phase.

Ingest of the VOD catalogues are handled by both “Metadata provisioning service” component and “Content Repository and Metadata” component.

The "Content Atmosphere" component and the “Content Similarity” component will be introduced as Specific Enablers in the upcoming release [4] of the Social Connected TV Platform.

3.1.4 - Personalised Media

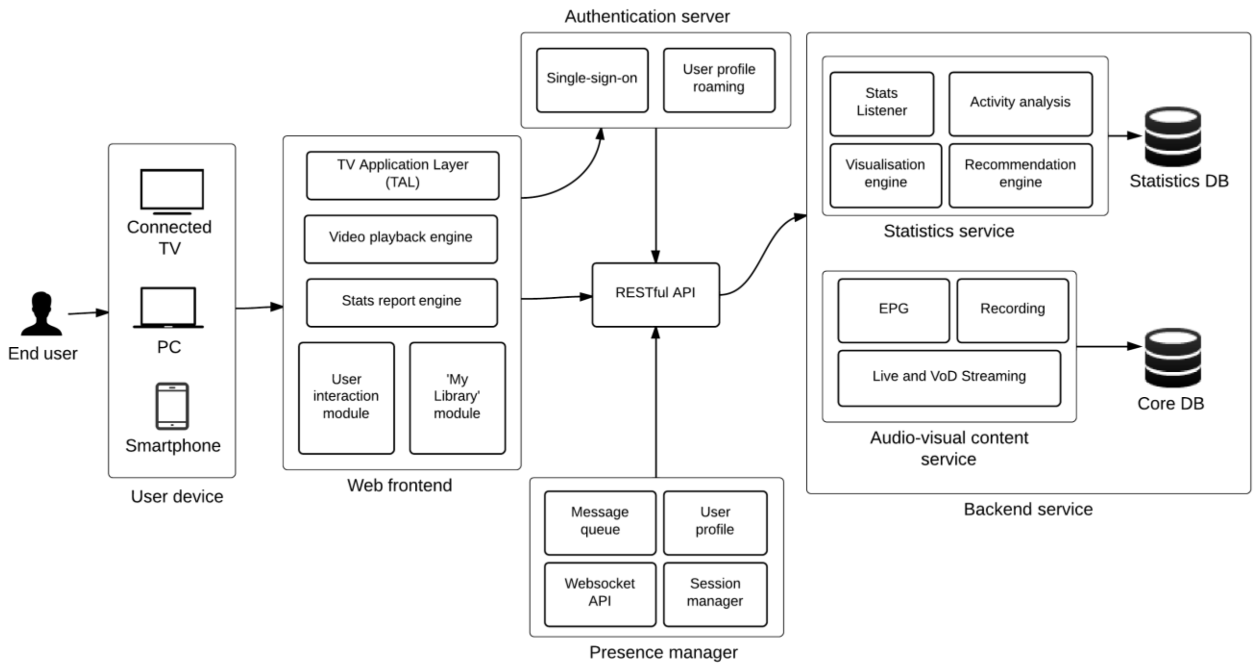


Figure 5 Realising applications in the area of Personalised Media.

Our goal in this scenario is to explore some of the different possible added values of personalised media in the context of an interactive IPTV service. We plan to introduce a series of features that allows users to manage their usage via different devices, and also to combine their consumption with that of others. One such feature is the Pause-Resume feature that allows users to view their watching history and resume playback regardless of the current or previous devices used. This is not only limited to devices capable of rendering the rich web interface, such as desktops, laptops, tablets, and smart phones), but also gaming consoles and smart TVs. This is made possible through an application we developed using the BBC’s TV Application Layer (TAL) SE (see Section 8.5) along with the DataCenter Resource Management (DCRM) GE (see Section 6.2.5).

The IPTV system provides programme information, live and on-demand TV/Radio streaming as its core service. Users can log in to the IPTV system on desktop, laptop, smartphone or tablet and view their TV [and radio] history including any programmes part-watched [and flagged for resume viewing], regardless of which device they last watched on. The single-sign-on and user profile roaming features of the authentication server support the log in function. The Stats report engine of the web frontend keeps track of play back position of a user session using heartbeat mechanism. Statistics service at the backend receives user stats reports from user devices via RESTful web service and provides features such as user activity analysis, visualisation and recommendation. After playback is paused on one user device (through the User interaction module), users can then resume play on a different device from the point at which they stopped viewing, [or from any other point]. Presence manager maintains user session, user profile and interacts with user devices for session transfer and synchronisation. A DCRM-based infrastructure is used to host the Presence manager, gathering their session information and watch history to facilitate seamless cross-device functionality. This provides tailored user experiences and allows the infrastructure to scale by creating a distinction between video content and user-specific data. TAL is used as a common framework to build applications for divergent platforms, such as smart TVs and gaming consoles. Suitable metadata is made available using the My Library module to the user about programmes displayed in this way so they can quickly identify what to watch [or listen to]. Users are also able to delete items from their history.

3.2 - Specific Enablers

We will provide the following Specific Enablers through the Social Connected TV Platform.

- Audio Fingerprinting SE (see Section 8.1) (Release 09/13)
- Audio Mining SE (see Section 8.2) (Release 09/13)
- Content Optimisation SE (see Section 8.3) (Release 09/13)
- Second Screen Framework SE (see Section 8.4) (Release 09/13)
- TV Application Layer SE (see Section 8.5) (Release 09/13)
- Content Atmosphere SE [4]
- Content Similarity SE [4]

We utilise the following common Specific Enablers for the Social Connected TV Platform.

- Content Enrichment SE (see Section 11.3)

3.3 - Generic Enablers

We take advantage of the following Generic Enablers from FI-WARE within the Social Connected TV Platform.

- DataCenter Resource Management (DCRM) GE (see Section 6.2.5)

We already actively use the following Generic Enablers as part of the Social Connected TV Platform.

- Identity Management GE (see Section 6.2.1)
- Semantic Annotation GE (see Section 6.2.2)
- Object Storage GE (see Section 6.2.4)

4 - Smart City Platform Architecture

The Smart City Platform enables creative people to generate, share and combine assets, objects and stories to develop mobility city services and new social experiences. Furthermore, it offers contextualization, recommendation, live information, mixed reality, 3D, sharing capabilities, and real-time communication. Moreover, the Smart City Platform contains a social interaction dimension where people can connect and build communities to exchange information via social networks.

The Smart City Platform is accessible for web and application developers and provides dedicated features to easily create Smart City Services with well-known and established technologies. In addition to that, the platform takes advantage of eBiz services, such as the Design My App Portal together with the App Gen SE [5], to support non-developers by allowing them to create use-case-specific applications with their content and according to their needs. Thereby, we introduce a shortcut for creative people realizing their ideas without requiring programming skill.

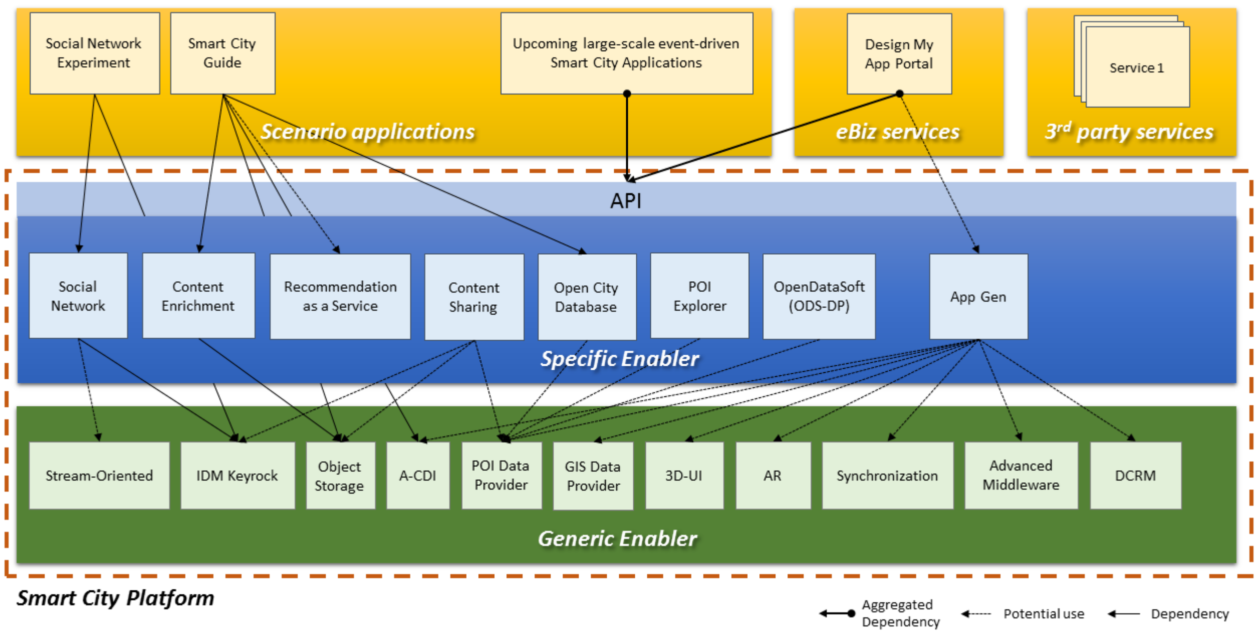


Figure 6 High-level architecture of the Smart City Platform

The Smart City Platform acts as an intermediate layer between the Generic Enablers provided by FI-WARE and the applications built on top of the platform. Further, it exposes the APIs of the Specific Enablers shipped with the platform to the application developers. A selection of applications showcase specific features or Enablers of the platform as illustrated in the figure above. Each application is dedicated to one scenario of the Smart City Platform as described in detail in deliverable D3.1. Moreover, we expose through the Design My App (DMA) Portal an additional service for 3rd-parties to take advantage of supported SEs within a few clicks by generating their own application based on our SEs.

Due to changes in the consortium, the focus of the Smart City Platform shifted slightly and the platform architecture was realigned during M12. In this context, we introduced several new concept in particular regarding the use-case-specific app generation and additional Specific Enablers as partial replacements. Thus, please treat this architecture description as a first draft that will be concreted before the next platform release in June 2014 (M18).

For previous versions of the Smart City Platform, please refer to previous versions of deliverables, such as D6.1 and D3.2. Moreover, the previous architecture description [6] is available in the FIcontent Wiki.

4.1 - Architecture Description

The Smart City Platform is built around the idea that users need easy and direct access to the services provided by FIcontent platforms. Consequently, we provide a portal to enable the user to choose from a set of Enablers and create apps and web apps directly from there. As such, the user can experience the novel functionalities of the services immediately and without needing to overcome a barrier of development effort. If the user needs more functionality that is not yet available by the service use cases, open APIs are provided to make it easy to access their interfaces and utilize the Enablers through programming.

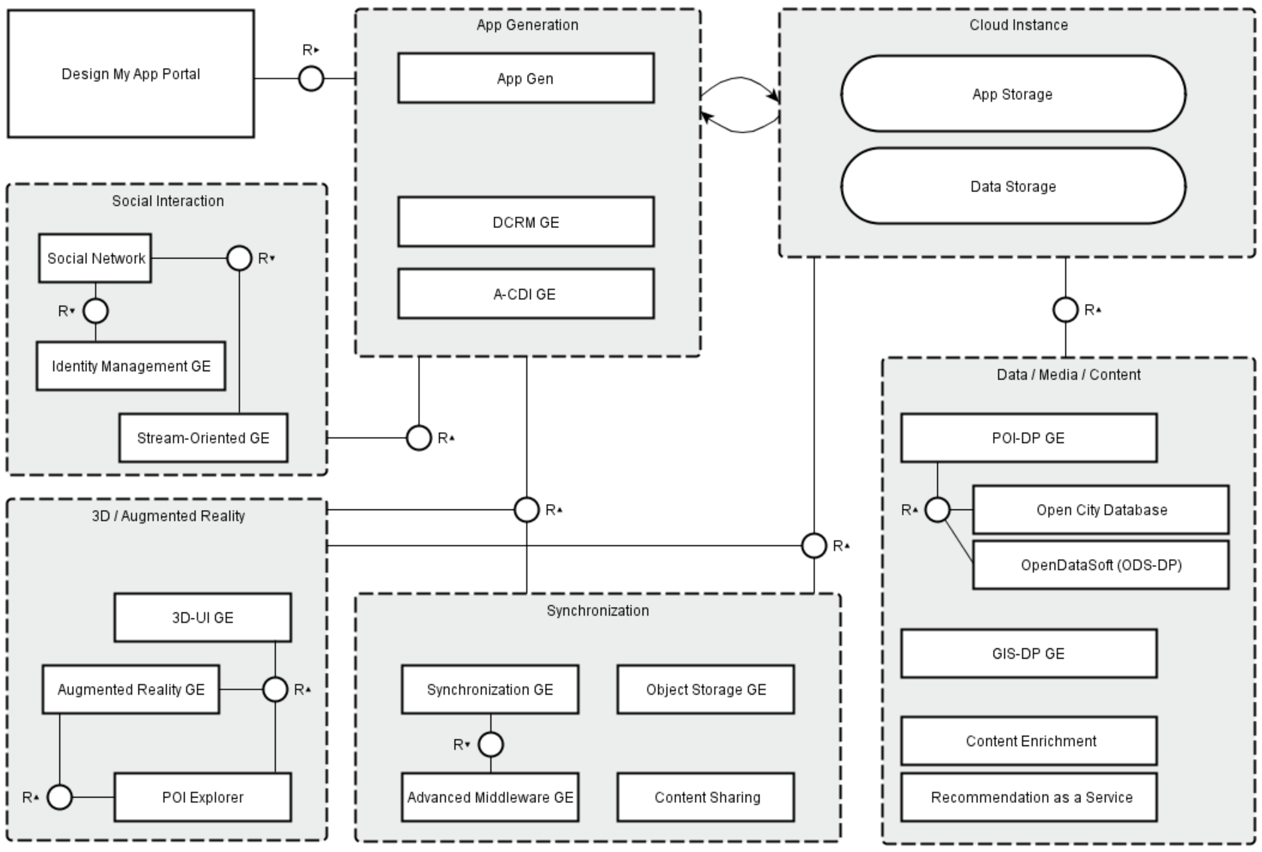


Figure 7 Architecture of the Smart City Platform including the interaction of SEs with GEs from FI-WARE

This architecture also addresses the needs of the different participant types. On one hand the **end user** can use the portal and create apps directly from the different enablers without development knowledge. Additionally, **event organizers** can enrich their applications with their own or third-party data from preconfigured formats. Since most of the Enablers have an open API, **developers** can dive deeper into the functionalities provided, and build their own feature sets on top of them. Last but not least, **platform developers and maintainers** can build rich web services with them and even add features to the app building platform, which makes it easy for Phase 3 parties to join the Smart City Platform.

4.1.1 - App Generation

To be able to create Smart City applications in a fast and convenient way, the architecture of the Smart City Platform has the app generation process as one of its core components. In its center works the App Gen SE [5] that creates apps and web apps. The interface to the user is driven by the Design My App Portal. The A-CDI GE provides a common API across a range of web-based runtime environments for mobile devices. Once the application is configured by the user, the DCRM GE spawns the server-sided services virtualized into the FI-Lab data center, for instance.

4.1.2 - Feature Building Blocks

On the lower left side of the diagram, the concrete services and functionalities are shown. For instance, the social interaction block consists of the Social Network SE (see Section 11.1) to create private, non-proprietary networks between end-users where they can share text and images. If the application does not have its own authentication system, we utilize the IDM-Keyrock GE to provide a ready to use OAuth2 compatible user management. If desired, the Stream-Oriented GE provides live video streaming between end users.

To enhance the visual experience of the Smart City Guide applications, we provide a 3D/Augmented Reality module. XML3D in the shape of the 3D-UI GE provides the capability to use 3D artifacts within your application. Moreover, these can be placed into the camera view by the Augmented Reality GE to turn your application into a full augmented reality experience.

4.1.3 - Data, Media and Content

Beside the functional components on the left side, we have possible data sources on the lower right side. Starting with the Data/Media/Content module, we have four main data sources to provide the capabilities of geo–localized services. The POI Data Provider GE and the Open City Database SE (see Section 9.1) are turn-key solutions to store and retrieve Points of Interest in the vicinity of the user. Furthermore, the OpenDataSoft (ODS-DP) SE [5] accumulates open data sources and turns them into structured, harmonized data sets. Including these and other possible data sources, the Content Enrichment SE (see Section 11.3) can create video annotations to enhance the multimedia experience of the applications. In addition, the Recommendation as a Service SE [5] provides a layer above the data sources to mine recommendations derived from the user behavior.

To be able to share data between multiple clients and enable real-time applications, we provide a Synchronization module. The Synchronization GE and Content Sharing SE (see Section 11.2) are able to share positioning details and generic data and replicate this data among different hardware. In addition, we use the Object Storage GE to store large-scale structured data into the virtualized cloud services of FI-LAB.

Overall, the app generation process supports the creation of virtualized Data Storage and App Storage instances that are orchestrated by the Design My App portal and is thereby a key feature of the Smart City Platform.

4.2 - Specific Enablers

We will provide the following list of Specific Enablers through the Smart City Platform.

- Open City Database SE (see Section 9.1) (Release 09/13)
- Recommendation as a Service SE [5]
- App Gen SE [5]
- OpenDataSoft (ODS-DP) SE [5]
- POI Explorer SE [5]

We will utilize the following list of common Specific Enablers for the Smart City Platform.

- Social Network SE (see Section 11.1) (Release 09/13)
- Content Enrichment SE (see Section 11.3) (Release 09/13)
- Content Sharing SE (see Section 11.2) (Release 09/13)

As a result of previous work from Orange on the Smart City Platform, two additional Specific Enablers are available, but no longer considered for the realigned architecture of the Smart City Platform.

- Recommendation Services SE [7] (Release 09/13)
- Virtual/Mixed Reality SE [8]

4.3 - Generic Enablers

We will take advantage of the following Generic Enablers from FI-WARE within the Smart City Platform. The specifications of some of them are in a very early stage. Thus, the actual functionality or the identifier may differ once the specification of the respective GEs is finalized.

- Stream-oriented GE (see Section 6.3.4)
- POI Data Provider GE (see Section 6.3.3)
- GIS Data Provider GE (see Section 6.3.6)
- 3D User Interface (3D-UI) GE (see Section 6.2.3)
- Augmented Reality GE (see Section 6.3.5)
- Synchronization GE (see Section 6.3.2)
- Advanced Middleware GE (see Section 6.3.1)
- DataCenter Resource Management (DCRM) GE (see Section 6.2.5)

We already actively use the following Generic Enablers as part of the Smart City Platform.

- Identity Management GE (see Section 6.2.1)
- Object Storage GE (see Section 6.2.4)
- Connected Device Interfacing (CDI) GE (see Section 6.2.7)

5 - Pervasive Games Platform Architecture

The architecture of the Pervasive Games Platform is a set of integrated, modular Enablers designed to aid building Internet-based games connected with the real world. While moving from native to in-browser execution as browser technology develops, we target real-time, low latency, and high performance goals with the Pervasive Games Platform. Moreover, the platform supports new forms of interactive entertainment of three tiers:

- **Tier 1 -- Digital Consumer Products (Toys).** This tier targets augmented-reality games based on toys, fashion, and other physical products. Games will use the product as a known and structured environment/level and include a limited number of networked uses, for storage and toy-to-toy communication.
- **Tier 2 -- Location Based Games.** Here, we target games developed in an installation such as a museum in which connected, cooperative game experiences are used to make the visit more compelling. This tier builds upon the first tier with real-world locations and a greater number of users.
- **Tier 3 -- City Wide Games.** The third tier targets city-wide games in which larger numbers of players interact in unstructured environments. Testing will take place in large-scale experimentation sites such as Zurich, Barcelona and Cologne. This tier is the most challenging as it requires a high degree of mobile connectivity as well as game dynamics implemented in unstructured locations.

The Pervasive Games Platform is accessible for web developers and provides dedicated features to easily create web-based games with well-known and established technologies. In addition to that, the platform takes advantage of the Unity native engine plugin to support professional game developers by allowing them to work with their accustomed tools.

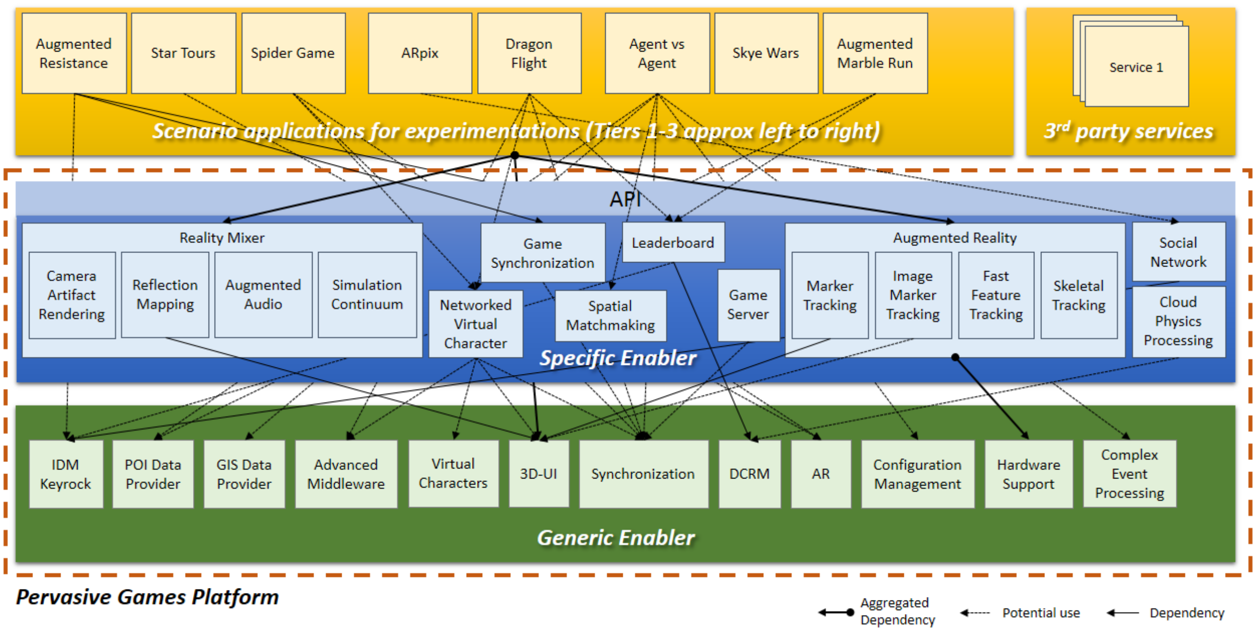


Figure 8 High-level architecture of the Pervasive Games Platform

The Pervasive Games Platform acts as an intermediate layer between the Generic Enablers provided by FI-WARE and the applications built on top of the platform. Further, it exposes the APIs of the Specific Enabler shipped with the platform to the application developers. A selection of applications showcase specific features or Enablers of the platform as illustrated in the figure above. Each application is dedicated to one scenario of the three tiers supported by the Pervasive Games Platform. The scenarios are described in detail in deliverable D4.1.

5.1 - Architecture Description

The core components of the Pervasive Games Platform are shaped by the Specific Enablers dedicated to domain-specific gaming scenarios. Thereby, the platform takes advantage of Generic Enablers from FI-WARE as well as common Specific Enablers developed in FIcontent. All of these Enablers may form groups, as shown in the figure below, and cover a range of features in game development, such as the group Augmented Reality SEs, which provides several tracking methods, or the group of Reality Mixer SEs, which focus on seamless augmented reality applications.

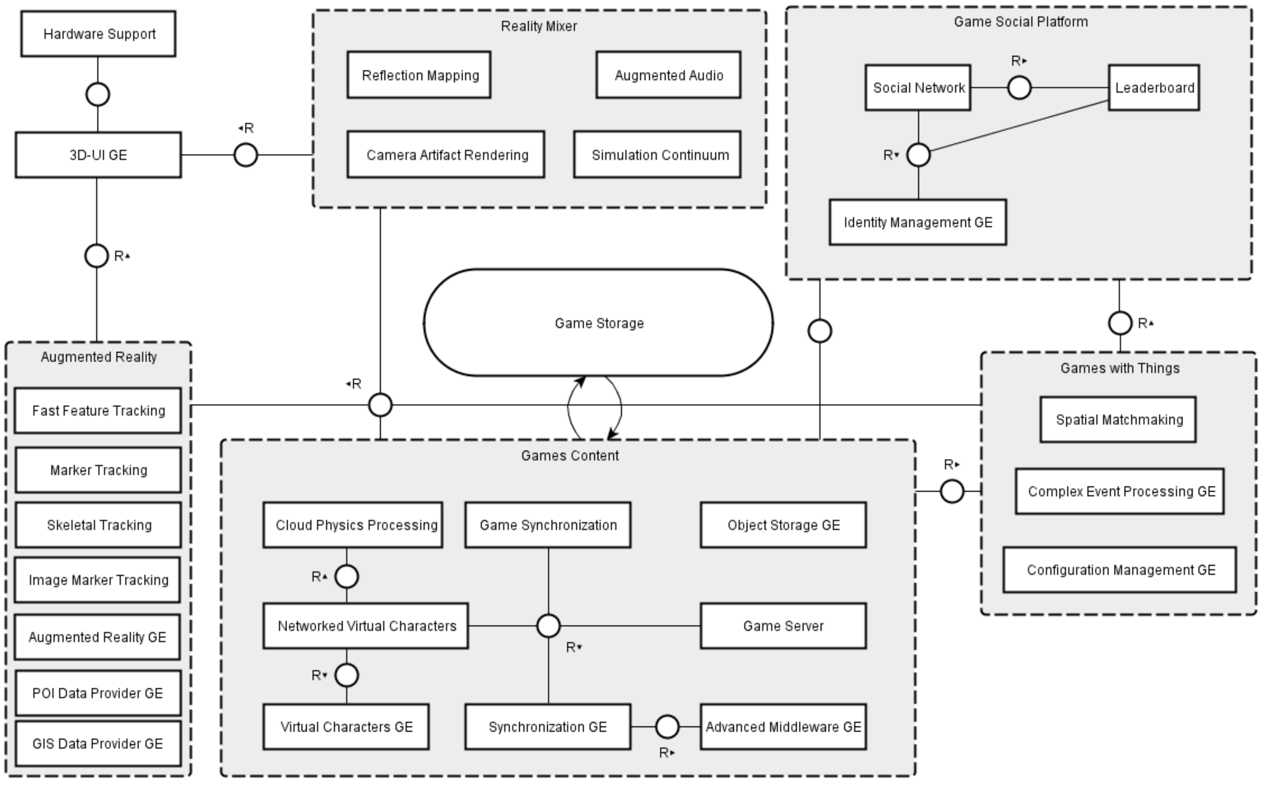


Figure 9 Architecture of the Pervasive Games Platform including the interaction of SEs with GEs from FI-WARE

The following sections describe the main functions provided by each component of the Pervasive Games Platform.

5.1.1 - Augmented Reality

Augmented Reality games break the line between reality and computer generated content by enhancing a real environment. The augmentation of a scene with additional computer generated content requires a very precise estimation of the camera pose. Therefore, we provide different tracking techniques based on the 3D User Interface (3D-UI) GE (see Section 6.2.3) in order to achieve a fully immersive experience in real-time. In addition to that, we utilize usual positioning methods, such as GPS localization, and provide with the Fast Feature Tracking SE (see Section 10.4) a GPU-accelerated implementation to easily create applications with basic markerless Augmented Reality functionality on today's mobile devices.

Moreover, this component incorporates the capabilities of the POI Data Provider GE (see Section 6.3.3) and GIS Data Provider GE (see Section 6.3.6) provided by FI-WARE to attach our content to existing objects in real-world environments. This becomes in particular relevant for Tier 3 applications together with the Game Synchronization SE (see Section 10.7).

5.1.2 - Reality Mixer

A context aware connected interactive experience must focus on the development of methods to integrate and match real or filtered video footage with rendered virtual objects and characters seamlessly. In this way the mobile device acts not as a traditional electronic display, but as a lens onto the real world with transparently aligned augmented reality content.

Moreover, we render augmented reality Computer Graphics (CG) objects that match the artifacts of the device's camera by utilizing the Camera Artifact Rendering SE (see Section 10.3). Adding synthesized computer images to live action footage, e.g. movie grain, noise, chromatic aberration and providing a reality slider that grades the disparity between video and CG footage achieves a seamless transition for mixed reality applications. In addition to that, stylization and beautification filters applied to both video stream and CG rendered content achieves an artistic goal avoiding the uncanny valley (as employed for example by PIXAR in traditional CG).

Augmented reality experiences further take place with virtual objects placed in the real world. Virtual objects under a physically accurate simulation thus far have no physical effect on real objects and conversely rigid and soft body dynamics captured from real objects have no physically simulated effect in virtual objects. This Enabler group includes methods for achieving a physical Simulation Continuum [9] between real and virtual objects.

Sound is also common in augmented reality productions. However, sounds are often not located at the correct projected 3D location. Furthermore, these do not account for the environment's aural signature, binaural effects, the influence of the presence of virtual objects and materials as sound reflectors. For these purposes we will provide the Augmented Audio SE [9].

5.1.3 - Game Social Platform

We utilize social platforms for the Pervasive Games Platform in two ways - one is intended for game developers and the other one for players. For both we take advantage of the Social Network SE (see Section 11.1) in combination with the Identity Management GE (see Section 6.2.1) from FI-WARE.

The developer portal is a useful place for game developers to find answers or help each other. It provides a starting point to get in contact with the capabilities of the Pervasive Games Platform and to receive information beyond the pure documentation, such as a Wiki including a FAQ section and a forum. Users can rate each other's entries, like tutorials or published games, and send messages using that developer portal. A news section and an optional mailing list will be used to inform about changes and announce new features and Specific Enablers of the Pervasive Games Platform. Standard software will be used for this.

Within the player social platform a dedicated area for each game will be provided including a forum to answer questions of players and to provide support. Ideally, more experienced players will help new players. The Leaderboard SE (see Section 10.1) seamlessly integrates into this platform, showing overall high scores for individual games or levels. Some games may require a group of players, like collaborative or competitive games. Therefore, the Spatial Matchmaking SE (see Section 10.6) connects to the social platform and helps to find nearby teammates.

5.1.4 - Games Content

This component of the platform is designed to provide services to manage the actual content of games. It allows the synchronization of UI and game states between players and devices and offers an efficient API for scene interactions and rendering. Therefore, we take advantage of several Generic Enablers alongside with Specific Enablers for game-domain specific purposes. The storage of game content, such as level data, avatars and assets, will be handled by the Object Storage GE (see Section 6.2.4) to provide a distributed, robust and scalable storage solution.

The Synchronization GE (see Section 6.3.2) can be used to create new types of multi-user applications shared between multiple devices and can be combined with the Game Synchronization SE (see Section

10.7) for taking charge of specific synchronization of game states. These Enablers rely on the technologies of the Advanced Middleware GE (see Section 6.3.1) to update game settings and 3D-scenes in multiplayer scenarios across different platforms.

Moreover, the Networked Virtual Character SE [9] interacts with the Synchronization GE (see Section 6.3.2) by exchanging motion data of virtual characters and applying the changes to the scenes. Those services may also rely on Cloud Physics Processing [9] to computationally intense parts to be streamed to a mobile device and the Virtual Characters GE (see Section 6.4.2) in order to provide animated and controllable characters to interact with the scene.

***5.1.5* - Games with Things**

The Games with Things feature set acts as a bridge between the Pervasive Games Platform and the Internet of Things. This enables games to handle both worlds - the virtual and real world, with both affecting each other. For this purpose the Backend Configuration Management Orion GE (see Section 6.4.9) will be utilized. Furthermore, we take advantage of the Complex Event Processing GE (see Section 6.4.8) to simplify the logic required to talk to a set of things, but also to allow them to act as a group and generate events more complex than they can as individual Things.

5.2 - Specific Enablers

We will provide the following list of Specific Enablers through the Pervasive Games Platform.

- Reality Mixer - Reflection Mapping SE (see Section 10.2) (Release 09/13)
- Reality Mixer - Camera Artifact Rendering SE (see Section 10.3) (Release 09/13)
- Reality Mixer - Simulation Continuum SE [9]
- Reality Mixer - Augmented Audio SE [9]
- Augmented Reality - Marker Tracking SE (see Section 10.5) (Release 09/13)
- Augmented Reality - Fast Feature Tracking SE (see Section 10.4) (Release 09/13)
- Augmented Reality - Skeletal Tracking SE [9]
- Augmented Reality - Image Marker Tracking SE [9]
- Leaderboard SE (see Section 10.1) (Release 09/13)
- Game Synchronization SE (see Section 10.7) (Release 09/13)
- Game Server SE [9]
- Networked Virtual Character SE [9]
- Cloud Physics Processing SE [9]
- Spatial Matchmaking SE (see Section 10.6) (Release 09/13)

We will utilize the following list of common Specific Enablers for the Pervasive Games Platform.

- Social Network SE (see Section 11.1) (Release 09/13)

5.3 - Generic Enablers

We will take advantage of the following Generic Enablers from FI-WARE within the Pervasive Games Platform. The specifications of some of them are in a very early stage. Thus, the actual functionality or the identifier may differ once the specification of the respective GEs is finalized.

- Advanced Middleware GE (see Section 6.3.1)
- Synchronization GE (see Section 6.3.2)
- Object Storage GE (see Section 6.2.4)
- Augmented Reality GE (see Section 6.3.5)
- Virtual Characters GE (see Section 6.4.2)
- Complex Event Processing GE (see Section 6.4.8)
- Backend Configuration Management Orion GE (see Section 6.4.9)
- POI Data Provider GE (see Section 6.3.3)

- GIS Data Provider GE (see Section 6.3.6)

We already actively use the following Generic Enablers as part of the Pervasive Games Platform.

- 3D User Interface (3D-UI) GE (see Section 6.2.3)
- Identity Management GE (see Section 6.2.1)
- DataCenter Resource Management (DCRM) GE (see Section 6.2.5)

5.4 - External Game Development Tools

One goal of the Pervasive Games Platform is to connect to existing game development tools created by European companies that have been proven in the field and supported by large communities. In the first platform release, we provide connections to:

- Unity3D from **Unity Technologies** (Denmark)
- SmartFoxServer, by **gotoAndPlay()** (Italy)

This list will be expanded over the course of the next year. We are currently evaluating the work of **GameAnalytics** (Germany), for instance.

5.4.1 - Unity3D

Unity3D [10] is a Game Development IDE, that provides a feature-rich Game Engine as well as a production environment for the creation of game scenes and the management of game assets. Unity3D supports JavaScript and C# code. Code and content of the Pervasive Games Platform relevant to Unity3D is distributed in form of unity packages - containers that facilitate the import and export operation between unity users.

The unity assets store is a marketplace for unity packages. Our packages will be uploaded and be available to the whole Unity community. We think this is a powerful distribution mechanism with a large pool of potential users.

5.4.2 - SmartFoxServer

SmartFoxServer [11] is a multi-platform server-client solution for multiplayer games and applications. It provides a rich set of features and extensive documentation, and comes with free and commercial licensing options. In particular, SFS provides functionality to create lobbies where players can chat, create instances of a game and start a playing session.

The server supports extensions for a variety of tasks while utilizing JavaScript and Python as programming languages. The Pervasive Game Platform will provide a number of server extensions to integrate both Specific Enablers and Generic Enablers from the Future Internet technology. SFS instances will run at least in the Zurich experimentation site.

6 - USAGE OF FI-WARE GEs IN FICONTENT

This section mainly documents the usage of FI-WARE Generic Enablers in FIcontent. The architecture descriptions of the Social Connected TV Platform, the Smart City Platform and the Pervasive Games Platform briefly explained the used GEs of the several platforms. The following sections collect the utilization of a GE with regards to the platforms and to the SE, which take advantage of certain GEs. Within the documentation of a specific SE the relation between the SE and the GE will be explained in more detail.

6.1 - Overall Uptake of Generic Enablers from FI-WARE

The following table reflects the overall uptake of GEs from FI-WARE by Specific Enablers and applications of FIcontent. The actual degree of uptake is indicated by

- D - meaning we have already taken the GE into out Demo PoC
- U - meaning we have already taken the GE into consideration in our design
- E - meaning we plan to experiment with it and consider it based on results

according to the spreadsheet FI-WARE GEis Planned Usage and General Information [12]. *Specific Enablers and applications* in italic denote that they will use or may take advantage of a respective GE (resulting in an uptake of U or E). Furthermore, few partners are evaluating GEs, which are not available yet and thus not listed here.

GE	Uptake	SEs / Applications
FI-Ware Cloud	D	Audio Fingerprinting SE Content Sharing SE Spatial Matchmaking SE Content Optimisation SE Audio Mining SE Smart City Guide (Android App) Search & Discovery App *TV programme companion* *Content Enrichment Demo* *Agent vs Agent*
DataCenter Resource Management (DCRM)	D	Leaderboard SE Augmented Resistance *TV Application Layer SE* *App Gen SE* *Dragon Flight* *Cross-Device Resume Play* *Augmented Marble Run*
Object Storage	D	Content Optimisation SE Content Enrichment SE Content Enrichment Demo Smart City Guide (Web App) *Content Sharing SE* *Smart City Guide (Android App)* *TV programme companion*
Complex Event Processing	E	*Agent vs Agent*
Stream-oriented	E	*Social Network SE* *Social Network Experiment* *ARpix*

GE	Uptake	SEs / Applications
Location	E	*Content Sharing SE* *Smart City Guide (Android App)*
Semantic Annotation	D	Content Optimisation SE *Content Enrichment Demo*
Backend Configuration Management Orion	E	*Agent vs Agent*
Identity Management GCP	D	Audio Mining SE Content Optimisation SE Audio Fingerprinting SE *TV programme companion* *Content Enrichment Demo*
Identity Management Digital Self	U	*Second Screen Framework SE* *rbbtext* *ARD Mediathek* *ARD-EPG* *Content Enrichment Demo* *TV programme companion* *Search & Discovery App*
Identity Management Keyrock	D	Social Network SE Social Network Experiment *Leaderboard SE* *Content Sharing SE* *Dragon Flight* *Augmented Resistance* *ARpix* *Augmented Marble Run* *Smart City Guide (Android App)*
AccessControl	E	*Content Sharing SE* *Smart City Guide (Android App)*
Connected Device Interfacing (CDI)	D	Smart City Guide (Web App) *App Gen SE*
Advanced Middleware	U	*Networked Virtual Character SE* *App Gen SE* *Spider Game Demo* *Dragon Flight*
3D User Interface (3D-UI)	D	Augmented Reality - Marker Tracking SE Reality Mixer - Reflection Mapping SE Star Tours ARpix Spider Game Demo Augmented Resistance *Networked Virtual Character SE* *Augmented Reality - Image Marker Tracking SE* *App Gen SE* *Dragon Flight*

GE	Uptake	SEs / Applications
Synchronization	U	*Networked Virtual Character SE* *Game Synchronization SE* *App Gen SE* *Spider Game Demo* *Dragon Flight* *Augmented Resistance* *Agent vs Agent*
GIS Data Provider	E	*App Gen SE* *Agent vs Agent*
POI Data Provider	U	*POI Explorer SE* *Content Sharing SE* *App Gen SE* *OpenDataSoft (ODS-DP) SE* *Open City Database SE* *Smart City Guide (Android App)* *Smart City Guide (Web App)* *Augmented Marble Run* *Agent vs Agent*
Augmented Reality	E	*App Gen SE* *Star Tours* *Spider Game Demo*
Virtual Characters	E	*Networked Virtual Character SE* *Spider Game Demo* *Dragon Flight*

6.2 - Generic Enablers actively used by FIcontent

6.2.1 - Identity Management GE

This GE is used for user management and identification within the Social Connected TV Platform (see Section 3), the Smart City Platform (see Section 4), and will be used by the Pervasive Games Platform (see Section 5) too. Thereby, the following SEs and applications currently take advantage of this GE:

- Audio Mining
- Audio Fingerprinting
- Content Optimisation
- Social Network
- Social Network Experiment

For the Social Connected TV Platform, the Identity Management GCP GE is being used by Fraunhofer IAIS in order to provide user login capabilities to the current graphical web interfaces of Audio Mining SE (see Section 8.2), Audio Fingerprinting SE (see Section 8.1) and Content Optimisation SE (see Section 8.3). A current limitation of the GE implementation is the missing functionality in the admin GUI to have a look at users that have registered for the respective Specific Enabler and to change their user data.

Moreover, IRT plans to use the IDM-DS implementation by NSN to allow a personalised user experience in applications using their Second Screen Framework SE (see Section 8.4). A profound limitation of current IDM GE implementations is that the GUIs are not applicable for the presentation on TV screens and the navigation with a remote control. IRT works together with NSN on a HbbTV compliant version of IDM-DS to come up with this constraint.

For the Smart City Platform and especially the Social Interaction module, we needed a user authentication component, which can be realized by an identity manager. We evaluated the existing implementations of this GE and found the 'Keyrock' Identity Manager ideal for our needs, since it is a full OAuth2-compatible implementation. Pixelpark uses an instance of this GE deployed to FI-Lab for their Social Network SE (see Section 11.1). Moreover, the Identity Management GE might be used by FOKUS to authenticate users and services against the Open City Database SE (see Section 9.1) as well as applications on top of the Smart City Platform.

In the context of the Pervasive Games Platform, the Identity Management GE will provide the tool for which players can be authenticated and recognized over multiple sessions. Furthermore, ETHZ is planning to use this GE as an optional part of their Leaderboard SE (see Section 10.1) for access control, in particular for creating and deleting games.

The Identity Management GCP GE [13], the Identity Management Digital Self GE [14], and the Identity Management Keyrock GE [15] are all available in the FI-WARE catalogue.

6.2.2 - Semantic Annotation GE

This GE is used within the Social Connected TV Platform (see Section 3) to enrich multimedia content. Thereby, the following SEs and applications currently take advantage of this GE:

- Content Optimisation

With regard to the Social Connected TV Platform, Fraunhofer IAIS takes advantage of this GE for their Content Optimisation SE (see Section 8.3) to semantically enrich textual content. It can be used for English, Spanish, French and Portuguese text content. The Semantic Annotation GE is available within the FI-WARE catalogue [16].

6.2.3 - 3D User Interface (3D-UI) GE

This GE extends the current declarative, rich media content model of HTML5 to also include interactive 3D graphics, and is used by Pervasive Games Platform (see Section 5). Thereby following SEs and applications currently take advantage of this GE:

- Reality Mixer - Reflection Mapping
- Augmented Reality - Marker Tracking
- Star Tours
- Spider Game Demo

Within the Pervasive Games Platform DFKI utilizes this GE to provide an extensive experience of gaming scenarios on the web. It is used in several demos showcasing their SEs based on this GE, such as Reality Mixer - Reflection Mapping SE (see Section 10.2) and Augmented Reality - Marker Tracking SE (see Section 10.5).

Moreover, the 3D-UI GE will be used for the Smart City Platform (see Section 4) too in order to supply Augmented Reality functionality on the web.

Since the GE is not yet officially released, only a description of the respective Epic is available within the FI-WARE Wiki [17] and a description of the implementation XML3D [18].

6.2.4 - Object Storage GE

This GE is used within the Social TV Connected Platform (see Section 3), the Smart City Platform (see Section 4), and will be used by the Pervasive Games Platform (see Section 5) too. The following SEs and applications currently take advantage of this GE:

- Content Optimisation
- Content Enrichment
- Content Enrichment Demo

- Smart City Guide (Web App)

The Content Optimisation SE (see Section 8.3) from Fraunhofer IAIS utilizes this GE as part of the Social TV Connected Platform to store enriched textual content.

For the Smart City Guide, the Object Storage GE is used by Fraunhofer FOKUS at the Berlin experimentation site to upload and store user generated videos of POIs.

In addition to that, this GE will be used by the Pervasive Games Platform to store game objects when scalability is an issue. This mostly includes upcoming Tier 3 game scenarios (city-wide games, MMOs) where a large number of players requires access to objects.

This GE is provided by Intel and is available within the FI-WARE catalogue [19].

6.2.5 - DataCenter Resource Management (DCRM) GE

This GE is used within the Pervasive Games Platform (see Section 5), and will be used by the Smart City Platform (see Section 4) too. The following SEs and applications currently take advantage of this GE:

- Leaderboard
- Augmented Resistance

In the context of the Pervasive Games Platform, ETHZ used this GE to deploy an instance of their Leaderboard SE (see Section 10.1) and an installation of the Identity Management GE at FI-Lab. In that way, the DRCM was and still is tested. For the upcoming city-wide gaming scenarios as described in D4.1, we plan to extend the tests of this GE, to instantiate virtual machines acting as game servers on demand.

Moreover, the App Gen SE [5] of the Smart City Platform will take advantage of this GE to ease the deployment of web application or server-side components of mobile applications generated by end-users.

For this GE are two implementations available in the FI-WARE catalogue from IBM [20] and Intel [21].

6.2.6 - FI-WARE Cloud GE

The Social Connected TV Platform (see Section 3) and the the Pervasive Games Platform (see Section 5) are using the FI-WARE Cloud GE to deploy their Specific Enablers. Thereby, the following SEs and applications currently take advantage of this GE:

- Audio Mining
- Audio Fingerprinting
- Content Optimisation
- Spatial Matchmaking
- Content Sharing
- Search & Discovery App
- Smart City Guide (Android App)

For the Social Connected TV Platform, Fraunhofer IAIS deployed their three SEs into the FI-Ware Cloud and thus make them available to 3rd-party developers.

In the context of the Pervasive Games Platform, Gobo designed their Spatial Matchmaking SE (see Section 10.6) to be deployed in the FI-Ware Cloud. Other platforms are also possible, but not recommended.

6.2.7 - Connected Device Interfacing (CDI) GE

The A-CDI provides a reference implementation of the CDI GE. It is a JavaScript API initially supporting the Android platform. It will provide a common JavaScript API across a range of web application frameworks including, Webinos, Tizen, Web Browsers (HTML5), and PhoneGap. The GE is utilized by the Smart City Platform. Thereby, the following SEs and applications currently take advantage of this GE:

- Smart City Guide (Web App)

Fraunhofer FOKUS is using the A-CDI GE in their Smart City Guide web application on top of the Smart City Platform.

The GE is available in the FI-WARE catalogue [22].

6.3 - Generic Enablers currently under evaluation by FIcontent

6.3.1 - Advanced Middleware GE

The Advanced Middleware GE (Codename KIARA) will be used within the Pervasive Games Platform (see Section 5) and Smart City Platform (see Section 4). This GE provides a convenient interface across different platforms and architectures to perform efficient, scalable and secure communication between software components.

DFKI and DRZ utilize the provided libraries to update game settings and their 3D-scenes in multiplayer scenarios across different client platforms.

Since this GE is not yet officially released, only the documentation of the respective Epic of this GE is available within the FI-WARE Wiki [23].

6.3.2 - Synchronization GE

The Synchronization GE provides the (partial) synchronization of scene content between one or more application instances, such that any changes happening in one will be propagated to its peers via a suitable server. It will be used within the Pervasive Games Platform (see Section 5) to enable multiplayer experience in games and within the Smart City Platform (see Section 4) to enable real-time sensitive mobile services in the context of Smart City Services.

The Game Synchronization SE (see Section 10.7) from DRZ will take advantage of this GE to provide a full-fledged synchronization mechanism in particular for server-client-based communication in gaming scenarios.

Moreover, the Networked Virtual Character SE [9] from DFKI will be based on this GE to synchronize motion of virtual characters from input, animation sequences and physical simulations.

A description of the Epic of this GE is available within the FI-WARE Wiki [24].

6.3.3 - POI Data Provider GE

The POI Data Provider GE (POI-DP GE) specifies an extensible and standardized format for describing Points of Interest as well as service interfaces to query them using various way from various sources. This GE will be utilized by the Smart City Platform (see Section 4) to manage POIs and their user generated content. Moreover, this GE becomes important for the Pervasive Games Platform (see Section 5) in Tier 3 city-wide gaming scenarios.

For instance, the Open City Database SE (see Section 9.1) from FOKUS and the OpenDataSoft (ODS-DP) SE [5] will be compatible with this GE.

In addition to that, the POI Explorer SE [5] from DFKI will take advantage of this GE to retrieve information about the advanced-interaction techniques supported by a particular POI. This becomes relevant in the context of the Pervasive Games Platform too.

Since the POI-DP GE is not yet officially released, only a description of the Epic of this GE is available within the FI-WARE Wiki [25].

6.3.4 - Stream-oriented GE

This GE will be used within the Smart City Platform (see Section 4) to utilize features such as live streaming and streaming of a video stored in a personal cloud.

Pixelpark successfully tested the 'Kurento' Stream-oriented GE, but it was not deployed in the first experimentation cycle. According to the developers, they expect to enhancing it for further scalability mid-2014. Then it might become integrated into the Social Network SE (see Section 11.1) from Pixelpark.

6.3.5 - Augmented Reality GE

The AR GE provides a high-level service to applications that implements typical AR functionality. It will be used within the Pervasive Games Platform (see Section 5) and the Smart City Platform (see Section 4).

This GE supports a subset of the Augmented Reality - Marker Tracking SE (see Section 10.5) from DFKI. It is useful if the natural feature tracking for using more advanced image markers instead of fiducial markers is not needed in a given scenario.

Since the GE is not officially released, only the description of the Epic is available within the FI-WARE Wiki [26].

6.3.6 - GIS Data Provider GE

The GIS-DP GE provides the infrastructure for querying and obtaining both outdoor 3D GIS data (terrain with objects) as well as building interiors in order to facilitate placing other data in a 3D context. It will be used by the Smart City Platform (see Section 4) to visualize virtual city models, for instance. Moreover, the Pervasive Games Platform (see Section 5) might take advantage of this GE with regard to the mixed reality gaming scenarios.

DFKI is investigating the concrete capabilities of this GE and the possible integration with the Synchronization GE to be utilized for both, real-time applications in the context of Smart City Services and mixed reality applications in upcoming city-wide gaming scenarios.

Documentation of this GE is available within the FI-WARE Wiki [27].

6.3.7 - Location GE

This GE might be used within the Smart City Platform (see Section 4) and the Pervasive Games Platform (see Section 5). The geofencing functionality of the Location GE could be used to send an alert to the user when he is close to a POI.

To use the Location GE in a given geographical area, a database describing the mobile network (cellules) in the area must be provided. Idem for the WiFi hot spots. This raises a problem if we want to implement the Location GE for trials in different cities/countries. This GE could be used for demos instead (ex. geofencing functionality). Moreover, there is no SUPL V2 compliant mobile available at the moment and we need to use an Android application provided by Thales.

The Location GE is available within the FI-WARE catalogue [28].

6.4 - Generic Enablers planned or considered to be used by FIcontent

6.4.1 - Service Capability, Connectivity and Control (S3C) GE

This GE might be used within the Smart City Platform (see Section 4) to provide functionalities such as audio conferencing, generation of calls, click2call functionality, and social Calling (conference calls with my Facebook friends). This GE is provided by Deutsche Telekom and Orange and is available within the FI-WARE catalogue [29].

6.4.2 - Virtual Characters GE

This GE should provide a basic infrastructure that enables applications to easily create, animate, and interact with virtual characters. This might be useful for the Pervasive Games Platform (see Section 5). A description is available within the FI-WARE Wiki [30].

6.4.3 - Big Data Analysis GE

Player analysis is a very important part of all games, especially in the Social Gaming space. This is achieved by gathering large quantities of data about player behavior and continuously analyzing it to tweak game parameters. Thus, the GE might be utilized by the Pervasive Games Platform (see Section 5). Currently there are commercial products that perform this, and we used the Open Call as a way of attracting developments in this area. This GE is available within the FI-WARE catalogue [31].

6.4.4 - Cloud Rendering GE

This Enabler is used to render graphic components on a server in the cloud. The results and inputs are exchanged between the server and the client machine. The Pervasive Games Platform (see Section 5) will utilize this GE to perform computationally intense graphics application on mobiles. A description of this GE is available within the FI-WARE Wiki [32].

6.4.5 - Apps Marketplace GE

Especially for independent game developers the advertisement, presentation, user ratings and comments are important. This functionality is provided by the Apps Marketplace GE. Hence, the Pervasive Games Platform (see Section 5) will take advantage of this GE to attract more independent game developers due to the offered ecosystem of development and deployment as this enhances the platform effect. This GE is available within the FI-WARE catalogue [33]. Further documentation is available within the FI-WARE Wiki [34].

6.4.6 - Apps Store GE

For games that are not free, once a potential gamer has decided to buy a game the Apps Store GE can be used to handle the payment. Documentation about this functionality is available within the FI-WARE Wiki [35].

6.4.7 - Access Control GE

The GE allows to manage authorization policies, and based on those policies, provides authorization decisions for requests to REST APIs of other services. It might be used within the Smart City Platform (see Section 4) to provide a common authorization mechanism for Smart City Services. The GE is provided by Thales and is available within the FI-WARE catalogue [36].

6.4.8 - Complex Event Processing GE

The CEP GE analyses event data in real-time, generates immediate insight and enables instant response to changing conditions. While standard reactive applications are based on reactions to single events, the CEP GE reacts to situations rather than to single events. Thus, it might by useful for the Pervasive Games Platform (see Section 5) to map complex game sequences into a set of rules. The GE is provided by IBM and is available within the FI-WARE catalogue [37].

6.4.9 - Backend Configuration Management Orion GE

The GE is part of the IoT of FI-WARE and might be useful for the Pervasive Games Platform (see Section 5) in the context of Games with Things. Thereby, this GE allows to register context producer applications, e.g. sensors, discover context producers information, e.g. which sensors are providing data for a given entity, and being notified when changes on context information are available. The GE is provided by Telefonica and is available within the FI-WARE catalogue [38].

6.5 - Generic Enablers evaluated and not going to be used by FIcontent

6.5.1 - Service Mashup GE

Extensive documentation and support is provided by Deutsche Telekom. However, it is not possible to add new services to the service repository. Only the services provided by the owner can be used to create a service mashup. Therefore, we are not able to use this GE for the use case scenario of the Social Connected TV Platform (see Section 3).

6.5.2 - Application Mashup GE

A demo account was requested. However, this GE was developed mainly for GUI-based service mashups and therefore does not fit the use case scenario of the Social Connected TV Platform (see Section 3).

6.5.3 - Service Composition GE

As stated in the FI-WARE Wiki ("The sustainability of this Open Specification cannot be guaranteed due to internal changes in the project consortium.") this GE is not supported.

6.5.4 - 3D Web Services GE

This GE offers a Web-service API for the 3D-Internet and XML3D specifically. It enables remote applications such as simulation, sources of sensor data, and other modules to provide input to and interact with the 3D scene. A Description of the respective Epic is available within the FI-WARE Wiki [39]. However, this GE is not available.

6.5.5 - Interface Designer GE

Designing non-trivial UIs in HTML directly is difficult. Do support developers and make the approach easier to use and increase uptake in the community an interface design tools is needed. It should allow for visually creating 2D/3D UI components, define their behavior, arrange them in 2D and 3D, and allow them to be connected to an application. Since the current implementation of this GE is not more than a basic in-browser DOM editor, there is no additional value of using it over the built-in DOM editors of today's web browsers. Overall, the GE seems to be different than the respective Epic described in the FI-WARE Wiki [40].

6.5.6 - 2D-3D-Capture GE

The GE should provide the ability to capture a real 3D environment via any suitable sensor in an end user's device, such as one or more images, depth images, video sequences as described in the respective Epic within the FI-WARE Wiki [41]. The implementation of this GE supports the collection of tagged images and some sort of spatial search. The ability of capturing real 3D environment - which would be needed for user scenarios of the Pervasive Games Platform (see Section 5) - is not supported.

7 - DEPLOYMENT OF FICONTENT PLATFORMS

In FIcontent, an experimentation site is mainly defined by an identified community of end-users in a given geographic location. Some of the partners hosting FIcontent experimentation sites are also part of the XIFI project and host XIFI nodes, i.e. datacenters. This is the case of ImaginLab in Britanny and Fraunhofer FOKUS in Berlin.

Having end-users test applications requires software and content hosted in datacenters in addition to client-side devices (smartphones or connected TVs). However, those are accessed through the Internet and therefore the physical location of the datacenters is, in many cases, irrelevant to the end user. Only in some cases requiring a connection with a very low latency or a very large bandwidth, the use of a datacenter in the vicinity of the experimentation site is required.

According to the FI-PPP Architecture Board representatives of the XIFI project, the Cloud GEis in the different XIFI nodes are progressively being connected to FI-Lab, which means that the deployment of server-side enablers will be done through a common portal and a common identity management. As of April 2014, only the original FI-Lab datacenter in Seville, and a new XIFI node in Trento are available through the FI-Lab portal, but this is bound to evolve.

The general recommendation for the deployment of enablers in FIcontent is therefore to use FI-Lab, in any of the available regions. The FI-WARE Testbed has also been used in the past instead because of the quota limitations in FI-Lab.

In some specific circumstances, for latency, bandwidth or confidentiality reasons, the server-side enablers are not deployed in the FI-Lab / XIFI Cloud, but in specific partner infrastructure. For instance, in the Lancaster experimentation site which provides IPTV streaming services to Lancaster University students and staff, the contract with the content providers prevent the content to be stored elsewhere.

D6.5 describes for each scenario a diagram showing the tested application(s) with the used enablers (both SEs and GEs), including their deployment location. The actual or planned deployment location is provided when available in brackets per Enabler (either SE or GE) in the diagram for each scenario. Since the infrastructure and deployment plans may differ between the experimentation sites, we further distinguish here per site.

8 - Specific Enablers of the Social Connected TV Platform

The Social Connected TV Platform is a toolbox that offers powerful instruments to enhance connected TV services. Thereby, we particularly focus on Specific Enablers to support three application scenarios. First, we will provide a multi-screen interaction technique with an intuitive interaction for advanced TV services, more versatile content presentation across screens and a personalized TV experience. Second, we will support connected TV services tailored to single and multiple users with social interaction between users, a search and discovery of content as well as user tracking and privacy. Finally, we will visualize personal content consumption by tracking implicit and explicit user interaction and providing users simple control over personal data.

The features of the Social Connected TV Platform address developers as well as providers of connected TV services. In the following sections we will present the Specific Enablers provided through the Social Connected TV Platform.

8.1 - Audio Fingerprinting SE

8.1.1 - Introduction to the Audio Fingerprinting SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [42] attached.

8.1.1.1 - Audio Fingerprinting Core

- Computes Fingerprint for Audio Signal
- Identifies related Content in Database
- Shows constantly synchronised content
- Implementation via RESTful API and Android SDK [43]

8.1.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Audio Fingerprinting SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the Audio Fingerprinting SE features. You should also be familiar with:

- RESTful web services
- HTTP/1.1

8.1.1.3 - Provider of the Audio Fingerprinting SE

For technical information please contact:

- Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
- Mr Michael Eble
- Tel.: +49 22 41 14 34 06
- Mail: michael.eble@iais.fraunhofer.de

8.1.1.4 - API Change History

Revision Date	Changes Summary
Sep 4, 2013	initial version
Apr 25, 2014	minor updates regarding GE usage; minor reformatting

8.1.1.5 - How to Read This Section

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

8.1.1.6 - Additional Resources

For more details about the usage of the Audio Fingerprinting SE within the FIcontent platforms, please refer to the Overview of the FIcontent Architecture (see Section 2) and Social Connected TV Platform Architecture (see Section 3).

A demonstrator Android app can be downloaded from our website [44].

8.1.2 - Overview of the Audio Fingerprinting SE

The Specific Enabler “Audio Fingerprinting” targets at scenarios in the context of Second Screen Applications like “Multi-Screen Experience”. Therefore, the SE recommends matching content for second screen devices: It analyses an incoming audio signal (e. g. from a TV or a VoD stream), computes a fingerprint and checks that fingerprint against a database potentially containing the analysed content data. Finally, the service returns matching content either as links to a repository or as the content itself. The service can be implemented into Android-based applications.

Features:

- Computes Fingerprint for Audio Signal
- Identifies related Content in Database
- Shows constantly synchronised content
- Implementation via RESTful API and Android SDK

Generic Enablers:

- **FI-WARE Cloud GE** The Audio Fingerprinting SE is deployed in the FI-WARE Cloud to be used and evaluated by developers.
- **Identity Management GCP GE** The Audio Fingerprinting utilises the Identity Management GE in order to allow the identification of users that use this SE.

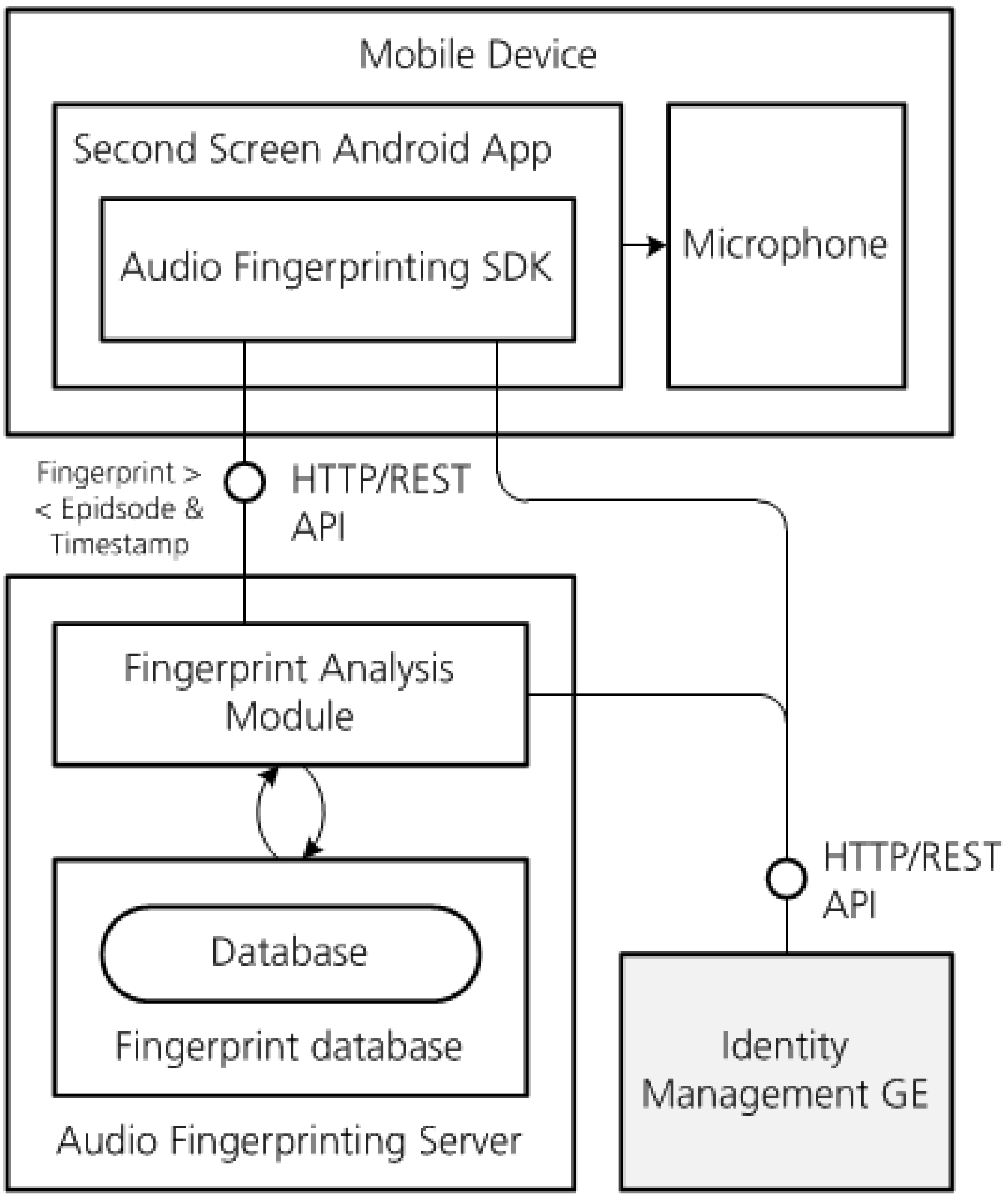


Figure 10 Audio Fingerprinting SE FMC diagram

8.1.3 - General Audio Fingerprinting SE API Information

In order to use the Fingerprint SE you need to get the iOS or Android Fingerprint SDK from Fraunhofer IAIS and integrate it into a mobile application. Alternatively, you can use Fraunhofer's example application available as APK file. Currently, there is no non-mobile, publicly available implementation that will record or load audio and calculate the necessary fingerprints for matching. Hence, the API definitions are currently only accessible over the iOS/Android SDK. The SDK is available for download here [43].

8.1.3.1 - Resources Summary

```
http://[server]/fingerprint
                             |--- /health              GET -> getHealth
                             |--- /?acode={acode}     POST -> matchFingerprint
```

8.1.3.2 - Authentication

Users need to be authenticated in order to access the Audio Fingerprinting administration and demo website. An account can be created using the Identity Management GE. The API does not support authentication in the current version.

8.1.4 - Audio Fingerprinting SE API Specification

Verb	URI	Description
"GET"	http:*[server]/fingerprint/health*	**getHealth** check if the Audio Fingerprinting SE is up and running
"POST"	http:*[server]/fingerprint/?acode={acode}*	**matchFingerprint** Match a precomputed audio fingerprint against the fingerprinting archive

8.1.4.1.1 - getHealth

Method	GET
Call	*http:*[server]/fingerprint/health
Path Parameter	None
Query Parameter	None
Headers	-
Body	-
Response	HTTP 200 if healthy
Sample Request	-

8.1.4.1.2 - matchFingerprint

Method	POST
Call	*http:*[server]/fingerprint/?acode={acode}
Path Parameter	None
Query Parameter	**acode**: Fingerprint archive code, automatically generated by the Fingerprinting SDK
Headers	Accept: application/xml or application/json
Body	Fingerprint, e.g., 3 707 90 7 1077 90 6 2202 9 ...
Response	HTTP 200 Response: asset ID
Sample Request	-

8.2 - Audio Mining SE

8.2.1 - Introduction to the Audio Mining SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [45] attached.

8.2.1.1 - Audio Mining SE Core

- Type of this SE: REST webservice
- Analyses TV content and turns speech into text
- Identifies segments and characteristic keywords
- Builds Index for Multimedia Search
- Generates recommendations based on video content (speech)
- Implementation via RESTful API

8.2.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Audio Mining SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the Audio Mining SE features. You should also be familiar with:

- RESTful web services
- HTTP/1.1

8.2.1.3 - Provider of the Audio Mining SE

For technical information please contact:

- Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
- Mr Michael Eble
- Tel.: +49 22 41 14 34 06
- Mail: michael.eble@iais.fraunhofer.de

8.2.1.4 - API Change History

Revision Date	Changes Summary
Sep 4, 2013	initial version
Apr 25, 2014	added recommendation API to documentation; minor updates regarding GE usage; minor reformatting

8.2.1.5 - How to Read This Section

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

8.2.1.6 - Additional Resources

For more details about the usage of the Audio Mining SE within the FIcontent platforms, please refer to the Overview of the FIcontent Architecture (see Section 2) and Social Connected TV Platform Architecture (see Section 3).

8.2.2 - Overview of the Audio Mining SE

The Specific Enabler "Audio Mining" targets at scenarios in the context of Multimedia Indexing and Search. Therefore, the SE analyses a given audio/video file and returns textual information for indexing. Speech and speaker segmentation as well as speech recognition are performed in order to turn speech into text. It delivers segments, characteristic keywords and various metadata. Finally, the SE builds an index for Multimedia Search.

Features:

- Analyses TV content and turns Speech into Text
- Identifies segments and characteristic keywords
- Builds Index for Multimedia Search
- Implementation via RESTful API

Generic Enablers:

- **FI-WARE Cloud GE** The Audio Mining SE is deployed in the FI-WARE Cloud to be used and evaluated by developers.
- **Identity Management GCP GE** The Audio Fingerprinting utilises the Identity Management GE in order to allow the identification of users that use this SE.

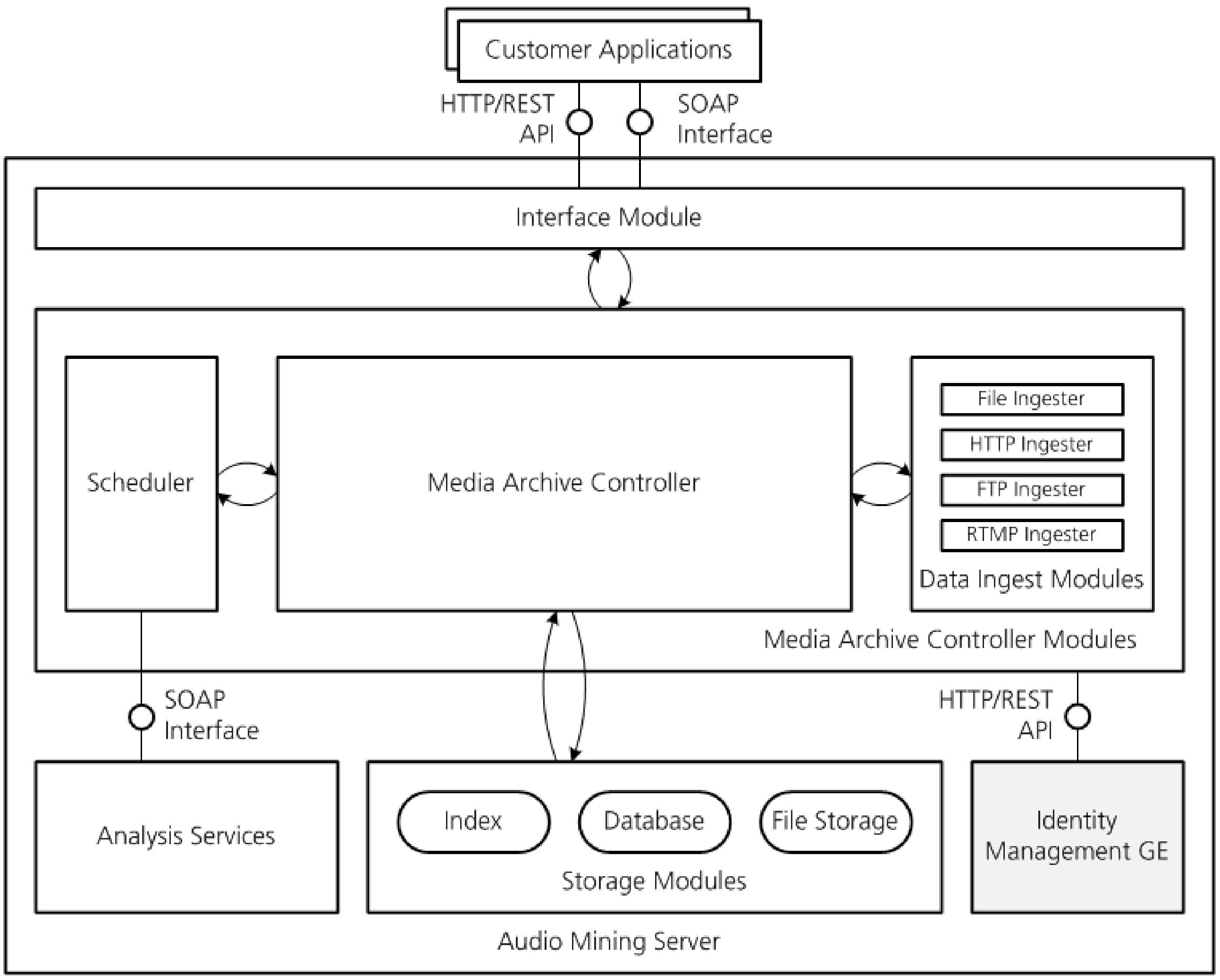


Figure 11 Audio Mining FMC diagram

8.2.3 - General Audio Mining SE API Information

This document describes the API methods to import new media assets into the Audio Mining SE. The Audio Mining API knows two types of objects: Indices and assets. Assets represent media objects including their manually and automatically created metadata whereas indices are collections of assets that are also stored and handled separately.

For more details about getting started with the Audio Mining SE within your own application or service, please refer to the Audio Mining SE Developer Guide [46] as a first starting point.The XML Schema definition (mediaArchive.xsd v1.2) [47] and a list of available speakers [48] for speaker identification is provided as well.

Resources Summary

```
http://<server>
       |--- /                                        POST -> createIndex
       |--- /{indexid}                               POST -> createAsset
                |                                    GET -> listAssets
                |                                    DELETE -> deleteIndex
                |--- /{assetid}                      GET -> getAssetMetadata
                |        |                           DELETE -> deleteAsset
                |        |--- /transcript            GET -> getAssetTranscript
                |        |--- /keywords              GET -> getAssetKeywords
                |        |--- /status                GET -> getAssetStatus
                |        |--- /structure             GET -> getAssetStructure
                |        |--- /subtitles             GET -> getAssetSubtitles
                |        |--- /recommendation        GET -> getRecommendations
                |--- /searcher-managed               POST -> search
```

8.2.4 - Audio Mining SE API Specification

Verb	URI	Documentation
"POST"	http:*[server]/*	**createIndex** Create a new index
"POST"	http:*[server]/{indexid}*	**createAsset** Create a new asset
"GET"	http:*[server]/{indexid}*	**listAssets** List all assets of an index
"DELETE"	http:*[server]/{indexid}*	**deleteIndex** Delete index and all assets in it
"GET"	http:*[server]/{indexid}/{assetid}*	**getAssetMetadata** Obtain plain asset metadata
"DELETE"	http:*[server]/{indexid}/{assetid}*	**deleteAsset** Delete a single asset
"GET"	http:*[server]/{indexid}/{assetid}/transcript*	**getAssetTranscript** Obtain asset transcript
"GET"	http:*[server]/{indexid}/{assetid}/keywords*	**getAssetKeywords** Obtain asset keywords
"GET"	http:*[server]/{indexid}/{assetid}/status*	**getAssetStatus** Obtain asset status
"GET"	http:*[server]/{indexid}/{assetid}/structure*	**getAssetStructure** Obtain asset segmentation structure
"GET"	http:*[server]/{indexid}/{assetid}/subtitles*	**getAssetSubtitles** Obtain asset subtitles in SRT format
"GET"	http:*[server]/{indexid}/{assetid}/recommendation*	**getRecommendations** Receive recommendations for a given asset
"POST"	http:*[server]/{indexid}/searcher-managed*	**search** Send a search request

8.2.4.1 - New Media assets

To make available a new media asset for search and synchronization you need to create an index first.

8.2.4.1.1 - createIndex

Method	POST
Call	*http:*[server]
Path Parameter	None
Query Parameter	**indexid**: Desired index name; **language**: Desired language ("German" only at the moment)
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 201 with body of ObjectUpdateResponseType
Sample Request	-

Afterwards you can trigger a new analysis for a media object available over HTTP, FTP, local disk path or as unprotected RTMP stream.

8.2.4.1.2 - createAsset

Method	POST
Call	*http:*[server]/{indexid}
Path Parameter	**indexid**: Index to be used
Query Parameter	None
Headers	Accept: application/xml or application/json / Content-Type: application/xml or application/json
Body	AnalysisRequestType encoded as XML or JSON String
Response	HTTP 201 with body of AnalysisResponseType
Sample Request	-

Currently the Audio Mining SE supports the following analysis types:

- Structural Analysis (finding audio segments and speaker turns, detect gender of speakers, diarize speakers)
- Speaker Recognition (finding segments with specific speakers)
- Speech Recognition (word and syllable-based German recognition only)
- Indexing for fingerprinting

A minimal AnalysisRequestType will look like this (see comments for details):

8.2.4.1.3 - AnalysisRequestType

Please refer to the Audio Mining SE Developer Guide [46]

8.2.4.2 - List asset information

To list information about available assets you have various options including listing all available assets of a single index as well as specific asset information such as the transcript or the keywords:

8.2.4.2.1 - listAssets

Method	GET

Call	*http:*[server]/{indexid}
Path Parameter	**indexid**: Index to be used
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 201 with body of AssetListResponse
Sample Request	-

8.2.4.2.2 - getAssetMetadata

Method	GET
Call	*http:*[server]/{indexid}/{assetid}
Path Parameter	**indexid**: Index to be used / assetid: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of Mpeg7
Sample Request	"GET http:*fi-content/ficontent/document1/"*

8.2.4.2.3 - getAssetTranscript

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/transcript
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of TranscriptResponseType
Sample Request	"GET http:*fi-content/ficontent/document1/transcript"*

8.2.4.2.4 - getAssetKeywords

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/keywords
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None

Response	HTTP 200 with body of KeywordResponseType
Sample Request	"GET http:*fi-content/ficontent/document1/keywords"*

8.2.4.2.5 - getAssetStatus

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/status
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of AssetStatusResponseType
Sample Request	"GET http:*fi-content/ficontent/document1/status"*

8.2.4.2.6 - getAssetStructure

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/structure
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of StructureResponseType
Sample Request	"GET http:*fi-content/ficontent/document1/structure"*

8.2.4.2.7 - getAssetSubtitles

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/subtitles
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: text/plain
Body	None
Response	HTTP 200 with body of text/plain
Sample Request	"GET http:*fi-content/ficontent/document1/subtitles"*

8.2.4.3 - Search

8.2.4.3.1 - search

Method	POST
Call	*http:*[server]/{indexid}/searcher-managed
Path Parameter	**indexid**: Index to be used
Query Parameter	None
Headers	Accept: application/xml or application/json / Content-Type: application/xml or application/json
Body	SearchRequestType encoded as XML or JSON String
Response	HTTP 200 with body of SearchResponseType
Sample Request	-

8.2.4.3.2 - SearchRequestType

Please refer to the Audio Mining SE Developer Guide [46]

8.2.4.4 - Deletion

Imported items can be deleted by calling a DELETE request on any index or asset resource. Keep in mind that the system does not have any question-response mechanism and deletions will apply instantly. Furthermore the deletion of an index will also delete all assets within this index.

8.2.4.4.1 - deleteIndex

Method	DELETE
Call	*http:*[server]/{indexid}
Path Parameter	**indexid**: Index to be used
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of ObjectUpdateResponseType
Sample Request	-

8.2.4.4.2 - deleteAsset

Method	DELETE
Call	*http:*[server]/{indexid}/{assetid}
Path Parameter	**indexid**: Index to be used; **assetid**: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of ObjectUpdateResponseType
Sample Request	-

8.2.4.5 - Recommendation API

8.2.4.5.1 - getRecommendations

Method	GET
Call	*http:*[server]/{indexid}/{assetid}/recommendation
Path Parameter	indexid: Index to be used / assetid: Desired asset
Query Parameter	None
Headers	Accept: application/xml or application/json
Body	None
Response	HTTP 200 with body of RecommendationResponseType
Sample Request	"GET http:*fi-content/ficontent/document1/recommendation"*

8.3 - Content Optimisation SE

8.3.1 - Introduction to the Content Optimisation SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [49] attached.

8.3.1.1 - Content Optimisation SE Core

- Type of this SE: REST webservice
- Enriches textual content by leveraging disperse sources
- Identifies persons, organisations and locations in textual content
- Delivers annotated content for faceted search
- Implementation via RESTful API

8.3.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Content Optimisation SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the Content Optimisation SE features. You should also be familiar with:

- RESTful web services
- HTTP/1.1

8.3.1.3 - Provider of the Content Optimisation SE

For technical information please contact:

- Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
- Mr Michael Eble
- Tel.: +49 22 41 14 34 06
- Mail: michael.eble@iais.fraunhofer.de

8.3.1.4 - API Change History

Revision Date	Changes Summary
Sep 4, 2013	initial version
Apr 25, 2014	major update of API documentation; minor updates regarding GE usage; integrated comments from internal review

8.3.1.5 - How to Read This Section

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

8.3.1.6 - Additional Resources

For more details about the usage of the Content Optimisation SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2) and the architecture of the Social Connected TV Platform (see Section 3).

8.3.2 **- Overview of the Content Optimisation SE**

The Specific Enabler “Content Optimisation” targets at scenarios in the context of Multimedia Mash-ups. Therefore, the SE processes incoming textual content (e. g. from Audio Mining SE) and extracts characteristic keywords. Afterwards, a semantic enrichment based on NLP will be performed to connect the transcripts and keywords with contextual information. Therefore, the SE integrates and harmonises additional content from disperse sources. The SE is intended for SMEs that want to build Second Screen Applications, e. g. for TV documentaries.Features:

- Enriches textual content by leveraging disperse sources
- Identifies persons, organisations and locations in textual content
- Delivers annotated content for faceted search
- Implementation via RESTful API

As a basic platform to provide most of the requirements, we will use a well-tested and frequently used software of Fraunhofer IAIS called Cortex. It combines a storage backend with a transformation workflow for XML-structured input and a customized and optimized Solr search index. The access to the internal data will be handled by a RESTful API interface.

We divide the functionality of our Cortex software into three parts:

- **Ingest.** The Ingest covers all functionality needed for pushing data into Cortex. The Ingest components perform transformation between XML formats and analysis of the content. The NER and further processing will take place in this step. The analyzed content will be stored in the Archive module.
- **Access.** The Access is responsible for retrieving data from the Archive and allows access to all components of the content like the source input data, the analyzed content and any further enrichment information.

- **Search.** The Search components are deeply integrated with a Solr search engine and provide the ability to search for specific terms and retrieve access information of corresponding assets from the Archive.

To fulfill all of the requirements we will integrate technologies of the University of Leipzig into our software platform. The research group Agile Knowledge Engineering and Semantic Web (AKSW) has developed a stack of semantic technologies to analyze textual content from disperse sources of the web. We are planning to integrate at least the following two technologies into our software platform:

- **DBpedia-Spotlight.** DBpedia Spotlight is based on an index of RDF information gathered from Wikipedia. All articles and their information have been preprocessed and combined in an collection of RDF data available at DBPedia.org. The RDF data has been used to create an language-dependent index for the retrieval of named entities by using statistical algorithms. Although the index is language-dependent, it is possible to setup indices of several languages to provide NER for those languages also.
- **FOX.** FOX is another technology based on several NER approaches. It allows a combined detection approach using several stand-alone technologies for NER but combining the results to reduce errors and increase accuracy.

We are going to integrate those technologies into our Ingest workflow. That means, we are plugging additional steps into our processing chain using the above mentioned services to perform an NER (and even some NLP) and embed the results into our stored data. This information will be available afterwards for our Search and Access.

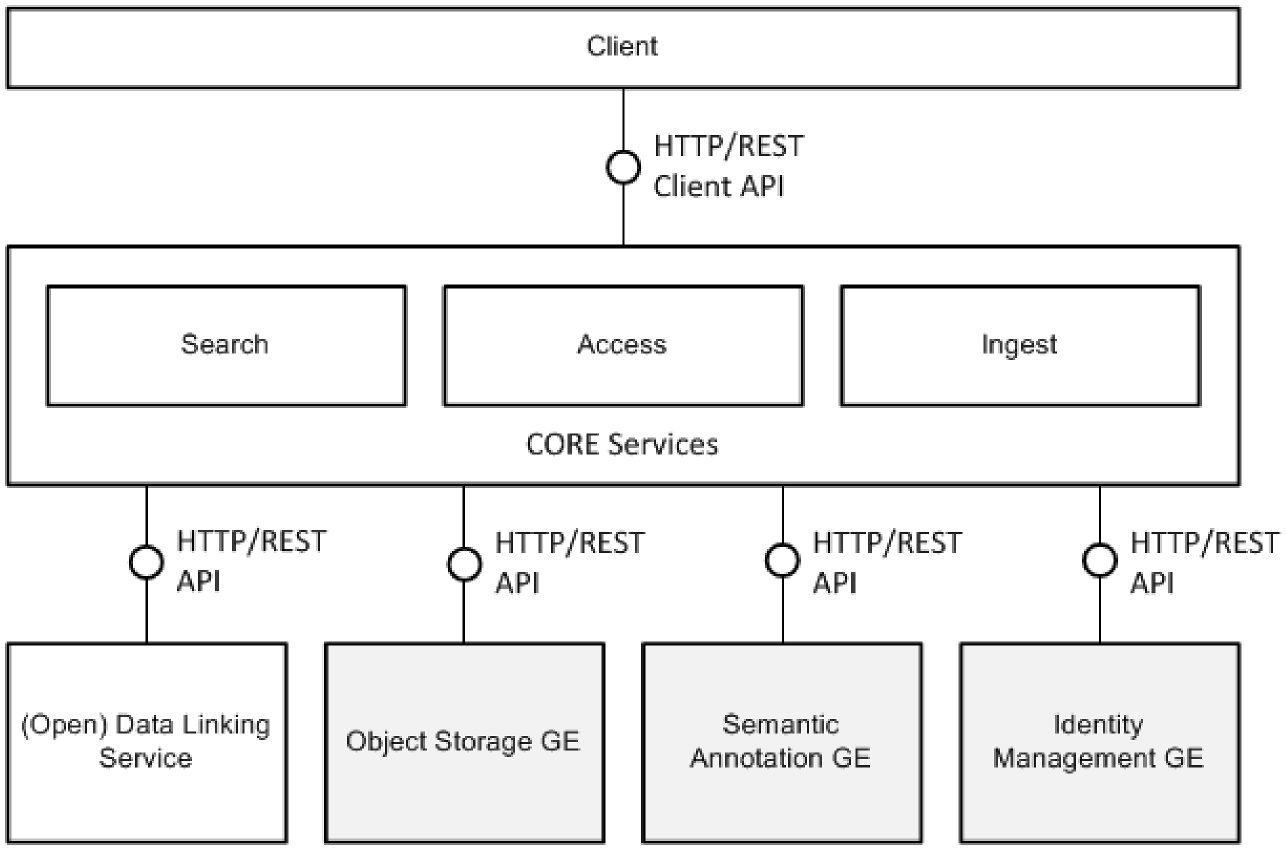


Figure 12 Content Optimisation SE FMC diagram

Generic Enablers

- **FI-WARE Cloud GE** The Content Optimisation SE is deployed in the FI-WARE Cloud to be used and evaluated by developers.

- **Identity Management GCP GE** The Content Optimisation SE utilises the Identity Management GE in order to allow the identification of users that use this SE.
- **Semantic Annotation GE** The Content Optimisation SE utilises the Semantic Annotation GE in order to perform Named Entity Recognition on English texts.
- **Object Storage GE** The Content Optimisation SE utilises the Object Storage GE in order to store semantically enriched data.

8.3.3 - General Content Optimisation SE API Information

For more details about getting started with the Content Optimisation SE within your own application or service, please refer to the Content Optimisation SE Developer Guide [50] as a first starting point.

Resources Summary

```
http://[server]/fc/annotation/annotate                         GET ->
getAnnotations

http://[server]:9996/ingest/{providerId}/{ingestId}/{revisionId}   PUT ->
importItem

http://[server]:8183/search/update                             GET -> commit

http://[server]:9998/
       |--- /search                                            GET -> search
       |--- /items/{itemId}                                    GET ->
getItem
       |--- /deleteItem/{itemId}                               DELETE ->
deleteItemById
       |--- /deleteProvider/{providerId}                       DELETE ->
deleteItemsByProviderId
       |--- /deleteIngest/{providerId}/{ingestId}              DELETE ->
deleteItemsByIngestId
```

8.3.4 - Content Optimisation SE API Specification

Verb	URI	Documentation
"GET"	http:*[server]/fc/annotation/annotate*	**getAnnotations** Receive Annotations
"PUT"	http:*[server]:9996/ingest/{providerId}/{ingestId}/{revisionId}*	**importItem** Import a new item
"GET"	http:*[server]:8183/search/update*	**commit** Commit changes to the search index
"GET"	http:*[server]:9998/search*	**search** Search the index
"GET"	http:*[server]:9998/items/{itemId}*	**getItem** Get item from database
"DELETE"	http:*[server]:9998/deleteItem/{itemId}*	**deleteItemById** Delete an individual item
"DELETE"	http:*[server]:9998/deleteProvider/{providerId}*	**deleteItemsByProviderId** Delete all items of a provider

"DELETE"	http:*[server]:9998/deleteIngest/{providerId}/{ingestId}*	**deleteItemsByIngestId** Delete all items of an ingest

8.3.4.1 - API

8.3.4.1.1 - getAnnotations

Method	GET
Call	*http:*[server]/fc/annotation/annotate
Path Parameter	None
Query Parameter	**text**: text to be annotated; **language**: [german,english,spanish,italian,portugese]
Headers	Accept: application/json
Body	None
Response	HTTP 200 with body of AnnotationResponseType
Sample Request	"GET http:*fi-content/fc/annotation/annotate?text=Angela%20Merkel&language=german"*

8.3.4.1.2 - importItem

Method	PUT
Call	*http:*[server]:9996/ingest/{providerId}/{ingestId}/{revisionId}
Path Parameter	**providerId**: ID of the provider; **ingestId**: ingest ID; **revisionId**: Current revision of this item. Trying to import an item for an existing revision results in an error
Query Parameter	None
Headers	Content-type: application/xml
Body	The body must contain the item to ingest. It must be compliant to the cortex.xsd schema "ItemType"
Response	HTTP 200
Sample Request	"PUT http:*[server]:9998/ingest/iais/123/1"*

8.3.4.1.3 - commit

Method	GET
Call	*http:*[server]:8183/search/update
Path Parameter	None
Query Parameter	**commit**: Can be either false or true. To perform a commit, it has to be set to true.
Headers	None
Body	None

Response	HTTP 200
Sample Request	"GET http:*[server]:8183/search/update?commit=true"*

8.3.4.1.4 - search

Method	GET
Call	*http:*[server]:9998/search
Path Parameter	None
Query Parameter	**query**: Term(s) to be searched. Must be URL-encoded and compliant to the SOLR Query Syntax. Default search value is "*:*"; **offset**: Number of the first entry of the result; **rows**: Count of result entries to be shown in total; **facet** (repeatable): Name of a facet which will be taken into account. Only facets which are supplied via a query parameter will be included into the result; **facet_name** (repeatable): Defines a search query on the given facet name. Allows to reduce the result by certain values of facets; **minDocs**: The amount of documents a facet must exceed to be included in the result set; **facet.limit**: Limits the number of values of a facet to the given amount; **sort**: Defines the sorting order (RELEVANCE, ALPHA_ASC, ALPHA_DESC, random_<seed>). Requires the query parameter otherwise the sort is always random_<seed>!
Headers	Accept: application/json
Body	None
Response	HTTP 200 with body of SearchResponseType
Sample Request	"GET http:*[server]:9998/search?query=Haus&rows=10&facet=place_fct&place_fct=Berlin&facet.limit=5&sort=ALPHA_ASC"*

8.3.4.1.5 - getItem

Method	GET
Call	*http:*[server]:9998/items/{itemId}
Path Parameter	**itemId**: The Id of the item returned by the search
Query Parameter	None
Headers	Accept: application/json or application/xml
Body	None
Response	HTTP 200 with body of ItemResponseType
Sample Request	"GET http:*[server]:9998/items/CJ2HFAG6JQZBFIITSYKOTDDHS3D2BS3R"*

8.3.4.1.6 - deleteItemById

Method	DELETE
Call	*http:*[server]:9998/deleteItem/{itemId}
Path Parameter	**itemId**: The id of the item to delete

Query Parameter	None
Headers	None
Body	None
Response	HTTP 200
Sample Request	"DELETE http:*[server]:9998/deleteItem/CJ2HFAG6JQZBFIITSYKOTDDHS3D2BS3R"*

8.3.4.1.7 - deleteItemsByProviderId

Method	DELETE
Call	*http:*[server]:9998/deleteProvider/{providerId}
Path Parameter	**providerId**: The id of the provider which items to delete
Query Parameter	None
Headers	None
Body	None
Response	HTTP 200
Sample Request	"DELETE http:*[server]:9998/deleteProvider/iais"*

8.3.4.1.8 - deleteItemsByIngestId

Method	DELETE
Call	*http:*[server]:9998/deleteIngest/{providerId}/{ingestId}
Path Parameter	**providerId**: The id of the provider which items to delete; **ingestId**: The id of the ingest of the provider
Query Parameter	None
Headers	None
Body	None
Response	HTTP 200
Sample Request	"DELETE http:*[server]:9998/deleteIngest/iais/123"*

8.4 - Second Screen Framework SE

8.4.1 - Introduction to the Second Screen Framework SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check the Terms and Conditions [51] for the Second Screen Framework SE.

8.4.1.1 - Second Screen Framework SE Core

- JavaScript library and API
- Full duplex communication between web applications running in the browser of connected TV devices and second-screen devices

- Mechanism for device discovery based on QR-code
- Persistent device connection -- devices once coupled remain associated
- Automatic launch of applications on the second-screen device
- Based on standardised web technology

8.4.1.2 - Intended Audience

The Second Screen Framework SE addresses developers and providers of browser based connected TV applications who want to add second-screen functionalities (cf. below (see Section 8.4.2)) to their apps.

8.4.1.3 - Provider of the Second Screen Framework SE

- Institut fur Rundfunktechnik GmbH
- Christoph Ziegler
- ziegler(at)irt.de
- +49 89 323 99-380

8.4.1.4 - API Change History

Revision Date	Changes Summary
Sept 16, 2013	initial version

8.4.2 - Overview of the Second Screen Framework SE

The Second Screen Framework SE provides web applications which are running on a TV with all the crucial functionalities to establish a persistent bi-directional communication path to a web application running in the browser of any second-screen device. This includes the possibility to launch applications from one TV on the second screen. All functionalities are provided via a lightweight JavaScript API and can thus be easily integrated into any web application.

Since the solution is fully compliant to the HbbTV standard it enables content providers to create fully interactive applications with direct programme relation potentially targeting millions of already deployed devices on the market. Thus, the concept can be implemented without modifications of hardware and only minimal extensions to existing applications.

8.4.2.1 - Understand the Second Screen Framework SE

The figure below illustrates the components of the Second Screen Framework SE. The framework components can be distinguished between those running on the server side and those running on the client side. Furthermore, the interfaces between the framework and the client application are highlighted.

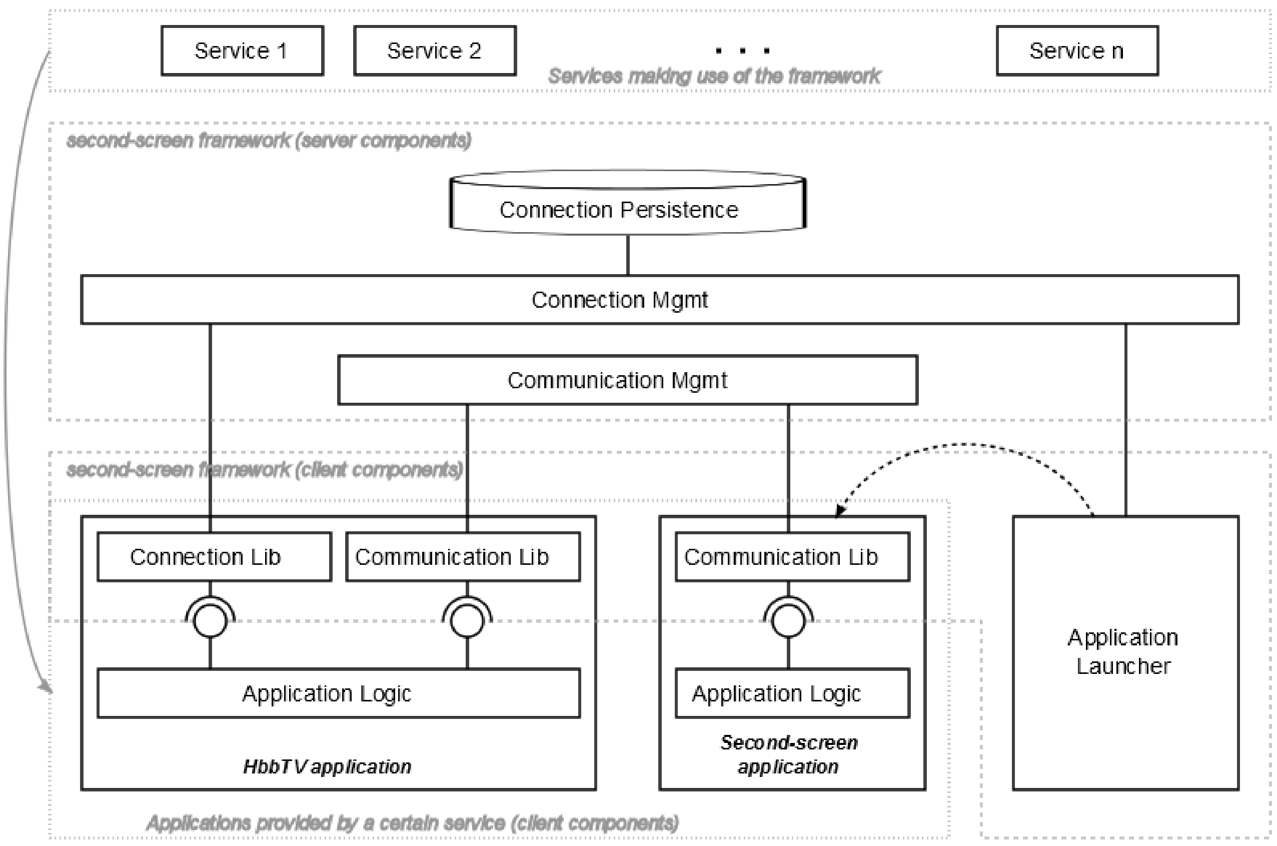

Service 1
Service 2
. . .
Service n
Services making use of the framework
second-screen framework (server components)
Connection Persistence
Connection Mgmt
Communication Mgmt
second-screen framework (client components)
Connection Lib
Communication Lib
Communication Lib
Application Logic
Application Logic
HbbTV application
Second-screen application
Application Launcher
Applications provided by a certain service (client components)

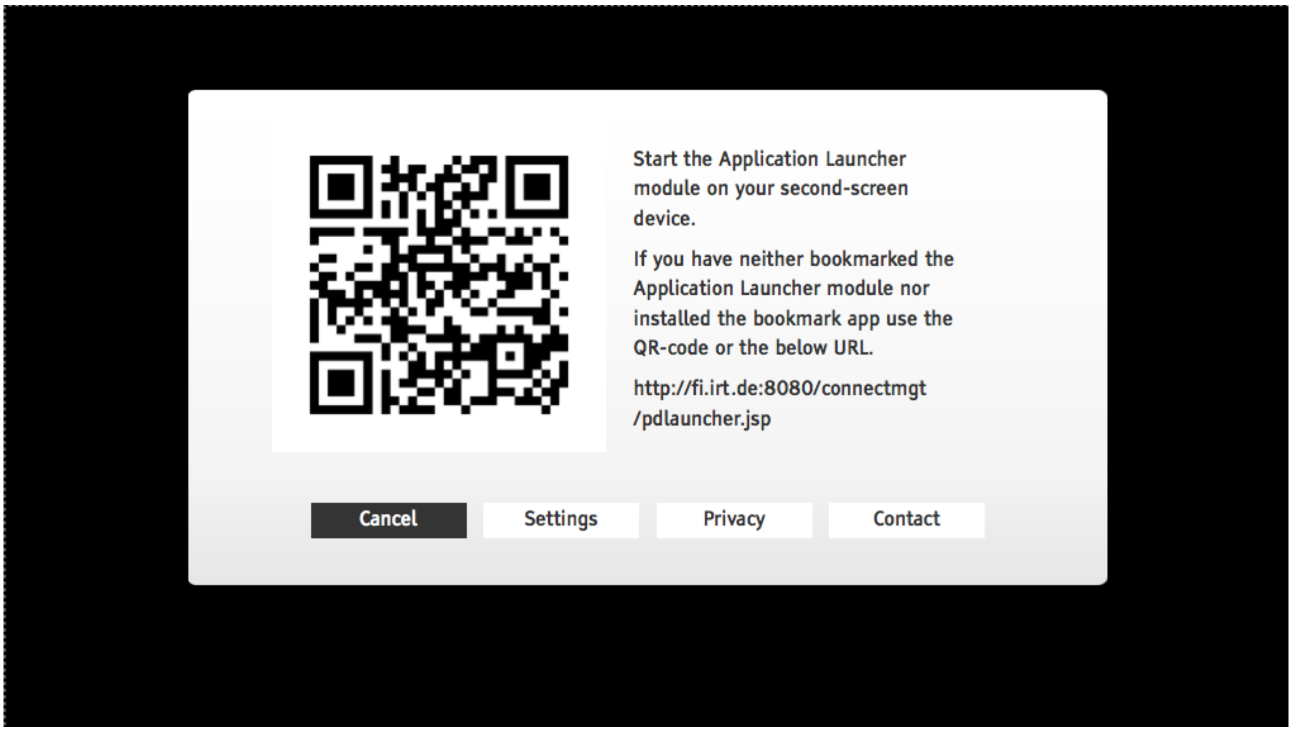
Figure 13 Second-Screen Framework SE component diagram

8.4.2.1.1 - Connection Mgmt (server)

The Connection Mgmt component at the server handles the device discovery and the device connection. It generates random but unique identifiers and generates the QR-code to be displayed on the TV device for the discovery process. It ensures that the second-screen device, that scans the QR-code and loads the encoded URL, is associated to the TV device. It makes sure that pairs of device IDs get persisted in the Connection Persistence so that connected devices remain associable and the discovery process needs to be executed only one time. Furthermore the Connection Mgmt server component provides the Connection Manager and the Application Launcher web application.

8.4.2.1.2 - Connection Manager (client)

The Connection Manager client component (not shown in the above architecture diagram) is an HbbTV compliant web application running in the browser of the TV device (cf. below screenshot of the Connection Manager application).



Start the Application Launcher module on your second-screen device.

If you have neither bookmarked the Application Launcher module nor installed the bookmark app use the QR-code or the below URL.

http://fi.irt.de:8080/connectmgt/pdlauncher.jsp

Figure 14 Screenshot of the Connection Manager application

The Connection Manager application handles the user interaction with the Framework server components and provides dialogues for multiple purposes, e.g. QR-code, disconnect devices, help texts etc. The Connection Manager application is launched when the appropriate API call is called by the web application using the Second-Screen Framework. The Connection Manager is typically launched when the user wants to connect a new second-screen device to its TV device.

8.4.2.1.3 - Connection Lib (client)

The Connection Lib client component is a JavaScript-library that provides wrapper classes in order to facilitate the communication with Connection Mgmt server component. It provides API calls in order to launch the Connection Manager application in the browser of the TV device and calls to request the connection status. All functionalities are provided by the PdManagement (see Section 8.4.5.2) class. Information on the connection status are wrapped in objects of type PdStatus (see Section 8.4.4.2.3).

8.4.2.1.4 - Application Launcher (client)

The Application Launcher component is responsible for the automatic launch of applications on the second screen. A screenshot of the Application Launcher application is given below.

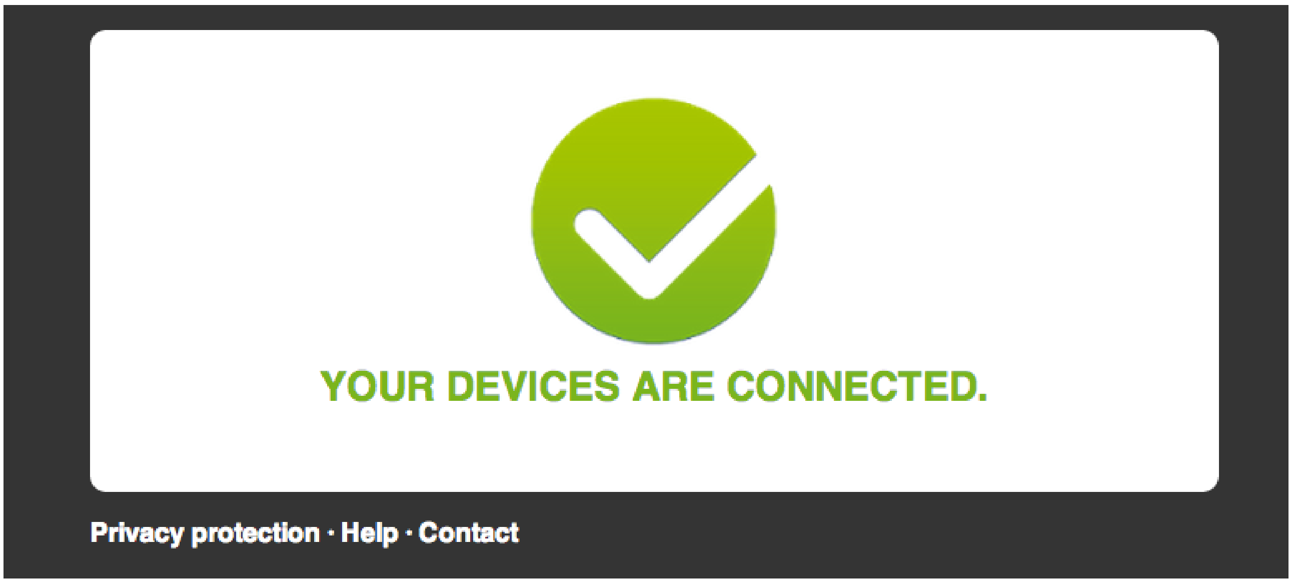

YOUR DEVICES ARE CONNECTED.
Privacy protection · Help · Contact

Figure 15 Screenshot of the Application Launcher application.

The Application Launcher application receives URLs to applications that are to be launched on the second-screen device. It is currently implemented as web application running in the browser of the second-screen device. The application should be running after the scanning of the QR-code and whenever an application is to be launched. When the Application Launcher receives a URL it automatically requests the browser to load the appropriate resource.

8.4.2.1.5 - Communication Mgmt (server)

The Communication Mgmt server component is responsible for the exchange of messages between connected devices. Clients send messages for connected devices to the Communication Mgmt server component. Clients register for messages from connected devices at the server. Whenever a message for a registered device is available, the server submits it to them.

8.4.2.1.6 - Communication Lib (client)

The Communication Lib client component is a JavaScript-library that provides wrapper classes in order to facilitate the communication with Communication Mgmt server component. It provides API calls in order to register for messages from connected devices and to send messages to connected devices. All functionalities are provided by the PdCommunication (see Section 8.4.5.1) class. Message information is provided by objects of type MessageEvent (see Section 8.4.4.2.1).

8.4.3 - Additional Resources

8.4.3.1 - Video Clip on the Second Screen Framework SE

This video clip [52] explains the principle of operation of the Second Screen Framework SE. It shows exemplary use-cases on the basis of two applications that make use of the SE.

8.4.3.2 - Develop applications with the Second Screen Framework SE

Find useful information on how to develop applications on the basis of the Second Screen Framework SE in the Developer Guide of the Second Screen Framework SE [53] section of this wiki.

8.4.4 - General Second Screen Framework SE API Information

8.4.4.1 - Objects

The Second Screen Framework provides its functionalities through methods of the following static objects:

- PdCommunication (see Section 8.4.5.1)
- PdManagement (see Section 8.4.5.2)

8.4.4.2 - Data Structures

8.4.4.2.1 - MessageEvent

"MessageEvent" is a container for events of the Communication Management (see Section 8.4.2.1.5) that contains the messages from the connected device(s) and information on the connection status.

```
var MessageEvent = {
    /*
     * The actual information that is to
     * be sent. 'message' is of type
     * {String}.
     */
    message: 'some message',

    /*
     * ID of the application that shall receive
     * the message. 'serviceId' is of type
     * {String}.
     */
    serviceId: 'sid:1000',

     /*
      * ID of the device that has send the
      * message. 'source' is of type
      * {String}.
      */
    source: '12345678',

    /*
     * Provides information on the status
     * of the connection to the Communication
     * Management. 'status' is of type {PdStatus}.
     */
    status: PdStatus,

    /*
     * ID of the device that is to receive the
     * message. 'target' is of type {String}.
     */
    target: '87654321'
};
```

8.4.4.2.2 - PdComStatus

Provides information on the status of the connection to the Communication Management Server (see Section 8.4.2.1.5).

```
var PdComStatus = {
    /*
     * Specifies an error that occurred when
     * 'PdComStatus' has been requested.
     * 'errorcode' is of type {Number}.
     *
     * Values:
     *   0 ... No error.
     *   11 ... Indicated device ID is not correct.
     *   12 ... Indicated service ID is not correct.
     */
    errorCode: 0,

    /*
     * Provides information on the status of the
     * communication channel. 'statusCode' is of
     * type {Number}.
     *
     * Values:
     *   0 ... Connection valid. The target device
     *         has registered a listener.
     *   1 ... Connection is not valid. The target
     *         device has no listener installed.
     *   2 ... Connection is not valid. Communication
     *         Management is not available.
     *   3 ... Connection is not valid. Target device
     *         is not available (e.g. HbbTV device is
     *         switched of).
     */
    statusCode: 0
};
```

8.4.4.2.3 - PdStatus

Contains information on the status of a certain device connection.

```
var PdStatus = {
    /*
     * Identifier of the device for whom the
     * {PdStatus} object was requested.
     * 'deviceId' is of type {String}.
     */
    deviceId: '12345678',

    /*
     * Specifies an error that has occurred
     * when the {PdStatus} object has
     * been requested. 'errorCode' is of
     * type {Number}.
     *
     * Values:
     *   0 ... No error.
     *   1 ... The used Device-ID could not be found
     *         or is not connected with the current device.
     *   2 ... Undefined error.
     */
    errorCode: 0

    /*
     * Statuscode indicating whether the call
     * launchPdApp() to launch an application
     * was successful (i.e. URL was launched on PD).
     * 'launchState' is of type {Number}.
     *
     * Values:
     *   0 ... Application was not launched or the call
     *         launchPdApp() was not performed.
     *   1 ... Application was launched successfully.
     */
    launchState: 0

    /*
     * Status code of the connection as managed by
     * the Connection Management Server. 'statusCode'
     * is of type {Number}.
     *
     * Values:
     *   0 ... An error occurred. The details can be seen
     *         in 'errorCode'.
     *   1 ... Personal Device is connected and the
     *         Application Launcher is active.
     *   2 ... Personal Device is connected but the
     *         Application Launcher is not running on the PD.
     */
    statusCode: 0
};
```

8.4.4.3 - Access to the JavaScript-libraries

In order to make use of the Second-Screen Framework SE's functionalities you have to integrate the following JavaScript resources:

- Connection Lib: https:*[server]/pdses/pdman.js into your connected TV application.*
- Communication Lib: https:*[server]/pdcom/pdcom.js into your connected TV and second-screen application.*
- JSONP library (for cross-origin messaging): https:*[server]/pdses/jsonp.js into your connected TV and second-screen application.*

Please contact the provider of the Second-Screen Framework SE (see Section 8.4.1.3) in order to resolve [server].

8.4.5 - Second Screen Framework SE API Specification

8.4.5.1 - PdCommunication

"PdCommunication" is an object which is realised as singleton and which makes available the functions of the Communication Management (see Section 8.4.2.1.5). The object is available immediately after initialising the framework.

8.4.5.1.1 - addMessageListener()

Registers an event handler for receiving messages.

8.4.5.1.1.1 - Parameters

- "messageHandler" "{Function}" is the callback function that is called when a message is send by the PD or the HbbTV device. The function is called with a parameter of type MessageEvent (see Section 8.4.4.2.1).
- "callback" "{Function}" is the callback function that is called when the status of the communication channel changes. The function is called with a parameter of type PdComStatus (see Section 8.4.4.2.2).
- ["source"] "{String}" is an optional parameter that indicates the source device of a message by the appropriate device ID.

8.4.5.1.2 - removeMessageListener()

Removes a certain message event handler.

8.4.5.1.2.1 - Parameters

- "messageHandler" "{Function}" is a reference to the callback function that has been referenced during the "PdCommunication.addMessageListener()" call.
- ["deviceId"] "{String}" is an optional parameter indicating the identifier of the device that is to be ignored.

8.4.5.1.3 - sendMessage()

Sends a message to a certain application running on a certain device. Messages are only passed if the sending device is connected to the device the message is addressed to.

8.4.5.1.3.1 - Parameters

- "callback" "{Function}" is a function that is called when the server response is available. The Function is called with a parameter of type PdComStatus (see Section 8.4.4.2.2). The parameter indicates whether the message was handled properly.
- "message" "{String}" contains the actual information to be delivered.

- "deviceId" "{String}" is the ID of the target device.
- "serviceId" "{String}" is the ID of the dedicated service.

8.4.5.2 - <u>PdManagement</u>

The "PdManagement" is an object which is realised as singleton and which makes the functions of the Connection Management Server (see Section 8.4.2.1.1) available. The object is available immediately after initialising the framework.

8.4.5.2.1 - getConnectedDevices()

Queries a list of devices connected to the calling HbbTV device.

8.4.5.2.1.1 - Parameters

- "callback " "{Function}" is called when the response of the server is available. The parameter of callback is an array of Strings which contains the device IDs of the devices that have a connection to the calling device.

8.4.5.2.2 - getDeviceId()

Queries the device ID of the calling device itself.

8.4.5.2.2.1 - Returns

- "{String}" Unique Identifier of the calling device.

8.4.5.2.3 - getPdStatus()

Queries the connection status of a specific Personal Device. The status is known by the Connection Management Server (see Section 8.4.2.1.1) and is retrieved from there.

8.4.5.2.3.1 - Parameters

- "callback " "{Function}" is a function which is called when the answer from the Connection Management Server (see Section 8.4.2.1.1) is available. The function is called with a parameter of the type PdStatus (see Section 8.4.4.2.3).
- "deviceId" "{String}" is the device ID of the device of whom the status is requested.

8.4.5.2.4 - launchApp()

Launches an applikation in the browser of the PD. The framework will append the Device-IDs of the PD and of the connected HbbTV devices to this URL . These IDs are relevant for using the Communication Management Server (see Section 8.4.2.1.5).

8.4.5.2.4.1 - Parameters

- "callback " "{Function}" is a function which is called if the Connection Management Server (see Section 8.4.2.1.1) gives back the result of the operation. The function is called with a parameter of the type PdStatus (see Section 8.4.4.2.3).
- "deviceId" "{String}" is the ID of the personal devices on which the URL is to be launched.
- "pdAppUrl" "{String}" is URL to be launched.

8.4.5.2.5 - launchPdManager()

Starts the Connection Manager Application (see Section 8.4.2.1.2). This draws the GUI for the device management as a full screen HTML page. The GUI is on top of a background image which shall be 1280x720 pixel. The URL of this background image can be handed over as a parameter. The GUI allows the user to do the first connection or new connections or asks him to start the Application Launcher (see Section 8.4.2.1.4). When connections are successfully managed or the user cancels actions the Connection

Manager Application (see Section 8.4.2.1.2) terminates and launches the return URL. The HbbTV applications should then request via the API of the Connection Manager Application (see Section 8.4.2.1.2) the new status.

8.4.5.2.5.1 - Parameters

- "returnUrl" "{String}" is the return URL started when the Connection Manager Application (see Section 8.4.2.1.2) is closed. This URL has to contain all status information the HbbTV application needs to continue.
- "backgroundUrl" "{String}" is the URL to the background image resource. The pixel size shall be 1280x720. Allowed are the picture formats specified by HbbTV.

8.4.5.2.5.2 - Returns

- "{Boolean}" Indicates whether the Connection Manager Application (see Section 8.4.2.1.2) can be loaded successfully.

8.5 - TV Application Layer SE

8.5.1 **- Introduction to the TV Application Layer SE**

The TV Application Layer (TAL) is an open source library for building applications for Connected TV devices. It is a software layer for developing connected TV applications that abstracts the differences between various connected TV, set-top box, and games console implementations, giving TV application authors a consistent API to develop against.

There are hundreds of different devices in the marketplace and they all use slightly different technologies to achieve the same result. The purpose of TAL is to allow you to write an application once, and be confident that it can be deployed to all HTML-based TV devices.

TAL is available to everyone under the terms of the Apache 2.0 open source license [54].

8.5.1.1 - Intended Audience

The TAL addresses developers of applications for connected TV devices, including broadcasters and independent application developers.

8.5.1.2 - Provider of the TV Application Layer SE

- British Broadcasting Corporation
- Chris Needham
- chris.needham@bbc.co.uk

8.5.1.3 - API Change History

Revision Date	Change Summary
Sept 18, 2013	Initial version

8.5.1.4 - Additional Resources

- Project homepage [55]
- Getting started guide [56]
- Code repository [57]

8.5.2 - Overview of the TV Application Layer

The TV Application Layer was developed as a way of simplifying TV application development. There are hundreds of different connected TV devices in the marketplace and they all use slightly different technologies to achieve the same result. The purpose of TAL is to allow you to write an application once, and be confident that it can be deployed to all HTML-based TV devices.

Our experience of writing TV Applications has shown us that while most Connected TV devices contain a web browser built upon something fairly familiar like WebKit or Opera, there are still some pretty big variations in the way that devices do the following: media playback, animation, networking, logging, JSON parsing, persistent storage, and remote control key codes. The TAL works to abstract these things.

Following is a component diagram of the TV Application Layer.

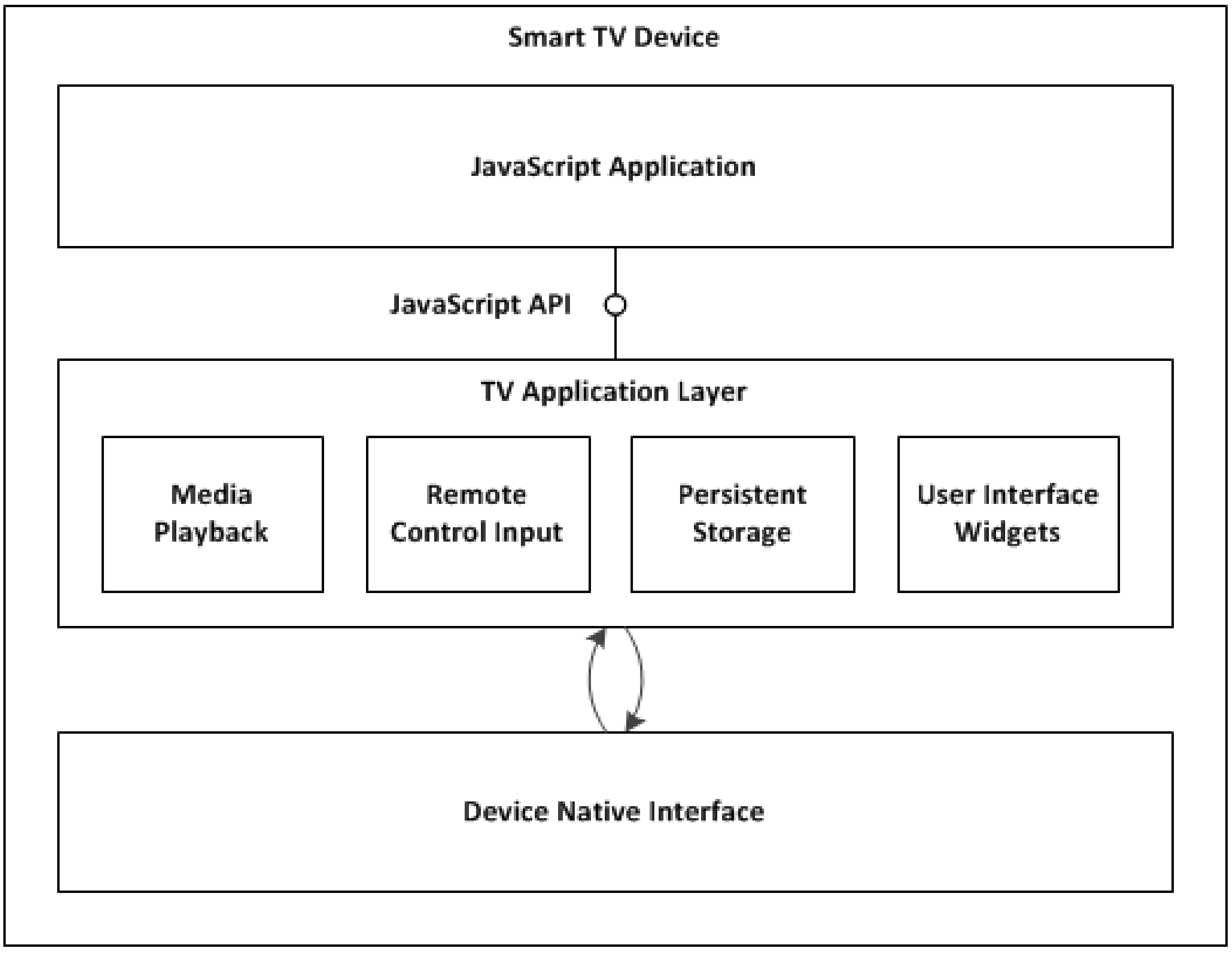


Figure 16 Component diagram of the TV Application Layer

8.5.3 - TV Application Layer API Specification

The TV Application Layer API is documented within the project's website: API reference [58].

9 - SPECIFIC ENABLERS OF THE SMART CITY PLATFORM

The Smart City Platform is a portfolio of functions, designed to foster the development and uptake of Smart City Applications based on Future Internet technologies. The Specific Enablers of the Smart City Platform will provide the technological foundation for Smart City Services to showcase dedicated features of the platform. Thereby, the applications will be available as Android native applications and as HTML5 web applications.

Due to the realignment of the Smart City Platform by M12, we drafted a new preliminary architecture of the Smart City Platform (see Section 4) including multiple new SEs for upcoming additional functionality. Thus, we consolidated the following section to present the currently available SEs, which are going to be used in the realigned Smart City Platform too. This affects only the Open City Database SE so far. More upcoming SEs, such as the App Gen SE [5], OpenDataSoft (ODS-DP) SE [5], and POI Explorer SE [5], dedicated to the Smart City Platform will be enclosed in future versions of D6.1 and other D3.x deliverables.

9.1 - Open City Database SE

9.1.1 - Introduction to the Open City Database SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions attached.

9.1.1.1 - Open City Database SE Core

- type of this SE: REST webservice
- collection of user generatet content for cities and POIs
- Data format JSON
- It is also possible to collects other information (pictures, videos, …) related to the touristic data from 3rd party providers such as Flickr, YouTube and Wikipedia

9.1.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Open City Database SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the TEMPLATE features. You should also be familiar with:

- RESTful web services
- HTTP/1.1
- JSON data serialization formats

9.1.1.3 - Provider of the Open City Database SE

- Robert Seeliger
- email: robert.seeliger@fokus.fraunhofer.de

9.1.1.4 - API Change History

adjust this paragraph respectivly

Revision Date	Changes Summary
Aug 22, 2013	initial version
Apr 16, 2014	second release

9.1.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

9.1.2 - Overview of the Open City Database SE

The main functionality that the Open City Database SE provides is:

- SPARQL REST-Endpoint of each OD platform
- External Resources: DBPedia, Online Access
- Cities
- POIs
- Users
- Comments
- Ratings
- Checkins

It provides a REST-API interface to the user.

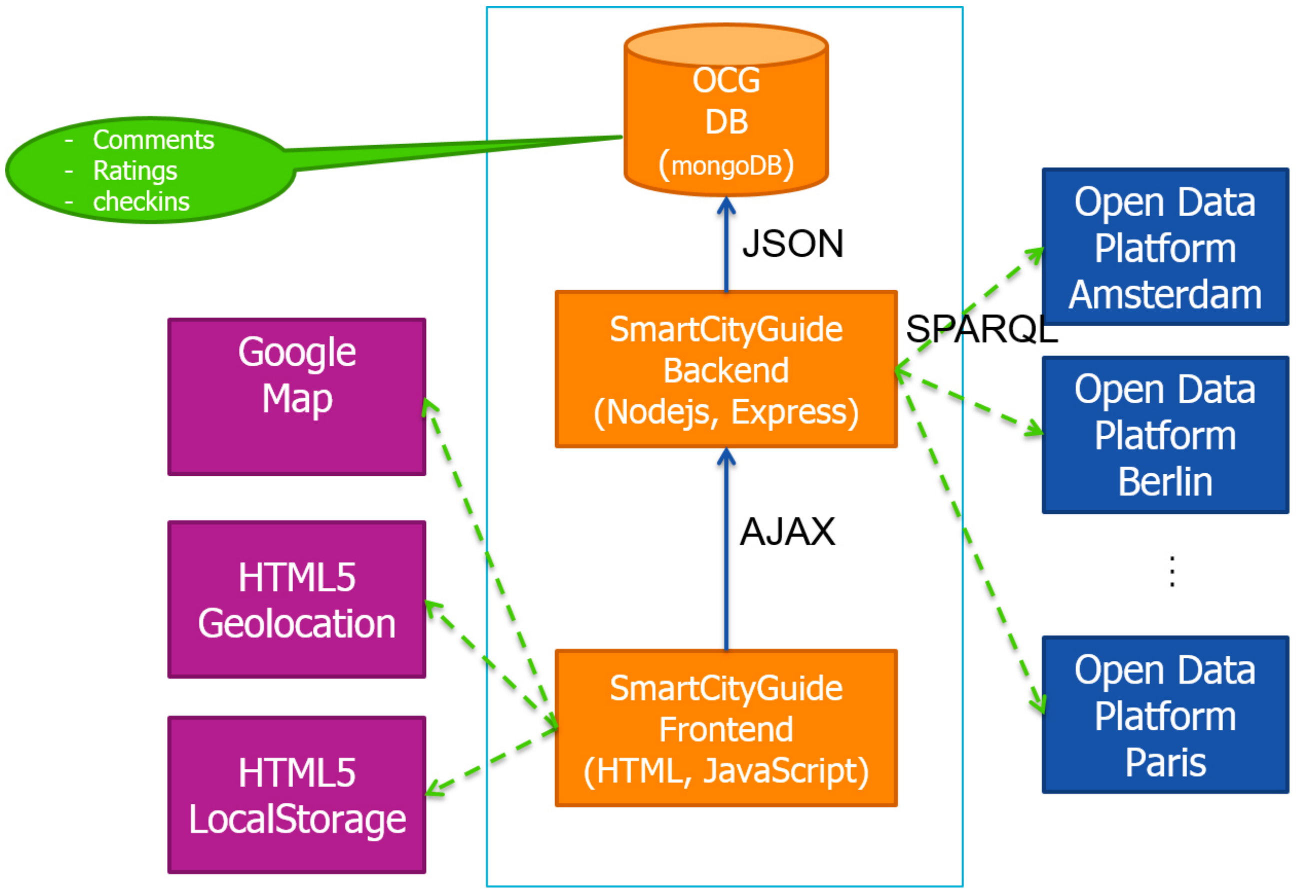

- Comments
- Ratings
- checkins
OCG DB (mongoDB)
JSON
SmartCityGuide Backend (Nodejs, Express)
AJAX
SmartCityGuide Frontend (HTML, JavaScript)
SPARQL
Google Map
HTML5 Geolocation
HTML5 LocalStorage
Open Data Platform Amsterdam
Open Data Platform Berlin
⋮
Open Data Platform Paris

Figure 17 Overview of the Open City Database components

9.1.3 - General Open City Database SE API Information

The goal of the Open City Database (OCDB) is to unify the acces s to open data platforms and portals while providing an easy programming interface.

The API described in this documentation is a stateless RESTful API and uses JavaScript Object Notion (JSON) as data transport format. It provides access to the content available in the OCDB and provides means to input user generated content (UCG).The underlying database is a document-oriented database, like mongoDB. Using this API data for different citiesand their Points of Interests can be retrieved for a usage with in Web applications.

Which data can be retrieved?

European City Data The OCDB includes some of the Europeans capitals with their name, country, latitude and longitude, a city description and attached media files, like an image.

Points of Interest Data Getting information about Points of Interest in a city. A POI object contains name, description, media files, latitude and longitude, contact information, opening hours, fee, city information and the category of thechosen POI.

9.1.4 - Open City Database SE API Specification

Please refer to the Open City Database API specification [59].

9.1.5 - Open City Database SE Developer Guide

REST API: GET Cities

```
{
   "country": "Germany",
   "image": "/assets/images/cities/coat_of_arms_berlin.png",
   "name": "Berlin",
   "_id": "4f4f4c27d4374e8001000007",
   "pois": {
      "total": 23
   },
   "location": {
      "latitude":52.500556,
      "longitude": 13.398889
      }
}
```

REST API: Create POI

```
{
   "name": "KaDeWe",
   "description": "Berlin’s most famous trademark department store is KaDeWe
(Kaufhaus des Westens) – or department store of the West. It is Berlin’s
shopping paradise, a favourite, easy to spot landmark on Wittenberg Platz. With
60,000sqm, the equivalent of nine football fields, 380,000 articles, 40,000
visitors a day, this is the legendary, largest department store on the
continent.",
   "image": "http://exampel.com/KaDeWe.jpg",
   "city": "5227507bec6eddda1900050d",
   "entrancefees": [],
   "openinghours": [
       "Mon-Thu 10-20, \nFri 10-21, \nSat 9.30-20"
   ],
   "contact": {
       "website": "http://www.kadewe.de/",
       "address": "Tauentzienstraße 21-24\n10789 Berlin",
       "telephone": [
           "Tel: +49 30 21210"
       ]
   },
   "publictransport": [
        "U-Bahn: Wittenbergplatz: U1, u2, U3"
   ],
   "location": {
         "latitude": 52.501666666667,
         "longitude": 13.341111111111
   }
}
```

10 - SPECIFIC ENABLERS OF THE PERVASIVE GAMES PLATFORM

In this section we present the Specific Enablers dedicated to the Pervasive Games Platform that are present in the first release of the platform. Notice how the name of the Enablers may contain the name of the group they belong to. This grouping is meant to facilitate the understanding of the overall goal of the Enablers. We consider the following areas:

- Augmented Reality: this group deals with the low-level technical challenges related to AR application, where tracking plays a central role.
- Reality Mixer: these Enablers deal with the problem of creating a proper illusion of presence for virtual elements as they are inserted into the real world. At the visual level, virtual objects should look as if they were real, not just with a proper placement and perspective but also with proper illumination and reaction to changes in the environment. Similarly, at the acoustic level, virtual sounds should behave like real ones, respecting the environment.
- Games with Things: this group draws from the concept of Internet of Things, but from a gaming perspective. Here the concept is to extend smart, connected, and sensory-rich objects with gaming capabilities.

10.1 - Leaderboard SE

10.1.1 - Introduction to the Leaderboard SE

Different Terms and Conditions might apply for each of the FIcontent Enablers.The Leaderboard SE is provided with the MIT License [60], see Terms and Conditions [61] for details.

10.1.1.1 - Leaderboard SE Core

- submit per-game high-scores for a player
- generate top lists of players per game
- two interfaces:
- REST Webservice
- Smartfoxserver 2X extension

10.1.1.2 - Intended Audience

This specification is intended for application developers. This document provides a specification of how to interoperate with the Leaderboard SE API. The reader should be familiar with:

- either RESTful web services
- or Smartfoxserver 2X development

10.1.1.3 - Provider of the Leaderboard SE

For technical information please contact:

- ETH Zurich, Switzerland
- Mr. Marcel Lancelle
- phone: +41 44 63 37 938
- email: marcel.lancelle(at)inf.ethz.ch

10.1.1.4 - API Change History

Revision Date	Changes Summary
Sep 15, 2013	fix APIv0.9 for FIcontent platform release 09/13

Revision Date	Changes Summary
Jan 31, 2014	added creation and deletion of games via REST interface

10.1.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., **void function(int a);**.
- An italic font is used to represent document titles or some other kind of special text, e.g., *STRING*.
- Variables are represented between brackets, e.g. {*id*} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.1.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Introduction to the Leaderboard SE (see Section 10.1).

For more details about getting started with the Leaderboard SE within your own application or service, please refer to the Developer Guide [62] as a first starting point.

***10.1.2* - Overview of the Leaderboard SE**

The main functionality that the Leaderboard SE provides is:

- Storage of high scores
- Retrieval of a sorted list of high scores

The Leaderboard SE is a combination of services plugged together to provide above functionalities. As such it is a construction of open source software solutions with their respective documentation found elsewhere.It provides a *type of API* interface to the user (developer).

In future, the Leaderboard SE will use the Identity Management GE to authenticate a user when submitting a score. The requests are a direct access to the database.

Beware that in this version, the highscore value to be submitted can easily be manipulated or hacked.

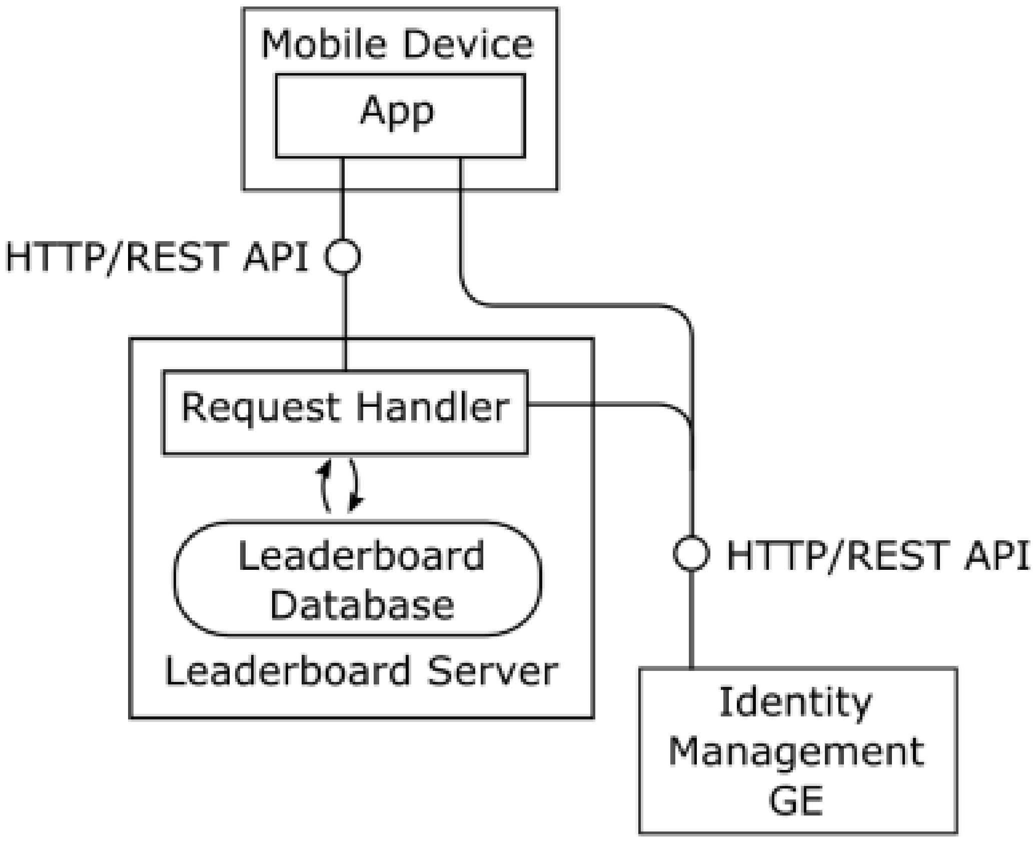


Figure 18 Leaderboard SE RESTful API

The diagram above shows the RESTful API, there is also a Smartfoxserver 2X extension available. The mobile device (or PC) can directly use the REST interface to interact with the leaderboard. In future, a simple access control is planned using the Identity Management GE.

10.1.3 - General Leaderboard SE API Information

Data types (pseudo notation):

```
ScoreEntry {
    string name     // e.g. "highscore"
    int value       // e.g. "10"
}
RankedListEntry {
    string playerID             // e.g. "gamer42"
    int rankingPosition         // e.g. "17"
    ScoreEntry[] scoreEntries   // e.g. {"name":"highscore", "value":"1025"}
    object userData             // e.g. {"comment":"Finally > 1000!!!"}
}
```

The authentication is not yet implemented, this is a preliminary version:

```
Authentication {
    string(?) playerIdentification // to verify identity of player with ID
Management GE from UPM
    string password             // for write access to database, to avoid simple
attacks, e.g. "1234"
}
```

10.1.4 - Leaderboard SE API Specification

10.1.4.1.1 - RESTful API Function Overview

All data is encoded in JSON.

HTTP method	URi	description	query parameters
"POST"	http:*[server]/{gameID}/{playerID}/score*	submit score	scoreEntries, userData
"GET"	http:*[server]/{gameID}/{playerID}/rankingposition*	get ranking position	orderBy, higherIsBetter
"GET"	http:*[server]/{gameID}/rankedlist*	get ranked list	rankStart, rankEnd, orderBy
"PUT"	http:*[server]/{gameID}*	create game table	-
"DELETE"	http:*[server]/{gameID}*	delete game table	-

10.1.4.1.2 - RESTful API Function Details

submit score

Method	POST
Call	*http:*[server]/{gameID}/{playerID}/score
Path Parameters	**gameID**: game name/ID, **playerID**: player name/ID, only a..z, A..Z, 0..9 allowed
Query Parameters	None
Headers	Accept: application/json
Body	**scoreEntries**: array of type *ScoreEntry*, a simple example is one element with name='highscore',value=999 , **userData** (optional): object, anything, e.g.: screenshot, comment, date/time
Response	HTTP 200 with body of type *RankedListEntry*
Sample Request	"POST http:*fi-content:4567/lb/myGame/gamer42/score"*

get ranking position

Method	GET

Call	*http:*[server]/{gameID}/{playerID}/rankingposition
Path Parameters	**gameID**: game name/ID, **playerID**: player name/ID
Query Parameters	None
Headers	Accept: application/json
Body	**orderBy**: string, which score to use for sorting if there are multiple scores, default is 'highscore', **higherIsBetter**: boolean, sorting order, the higher the score the better, default is 'true'
Response	HTTP 200 with body of type *RankedListEntry*
Sample Request	"GET http:*fi-content:4567/lb/myGame/gamer42/rankingposition"*

get ranked list

Method	GET
Call	*http:*[server]/{gameID}/rankedlist
Path Parameters	**gameID**: game name/ID
Query Parameters	**rankStart**: int, the rank of the best player requested, for top ten this would be '1', **rankEnd**: int, the rank of the worst player requested, for top ten this would be '10', **orderBy**: string, which score to use for sorting if there are multiple scores, default is 'highscore'
Headers	Accept: application/json
Body	None
Response	HTTP 200 with body with a list of type *RankedListEntry*
Sample Request	"GET http:*fi-content:4567/lb/myGame/rankedlist"*

create game table

Method	PUT
Call	*http:*[server]/{gameID}
Path Parameters	**gameID**: game name (only a..z, A..Z, 0..9 allowed)
Query Parameters	None
Headers	Accept: application/json
Body	None
Response	HTTP 200
Sample Request	"PUT http:*fi-content:4567/lb/myGame"*

delete game table

Method	DELETE
Call	*http:*[server]/{gameID}

Path Parameters	**gameID**: game name
Query Parameters	None
Headers	Accept: application/json
Body	None
Response	HTTP 200
Sample Request	"DELETE http:*fi-content:4567/lb/myGame"*

Note: for creation and deletion it is planned to have an access control via the Identity Management GE.

10.1.4.1.3 - SFS2X Extension call API Functions:

```
void submitScore(gameID, playerID, scoreEntries, authentication, userData)
```

gameID: string to identify the game
playerID: string to identify the player
scoreEntries: array of **ScoreEntry**, a simple example is one element with **name='highscore',value=999**
authentication: Authentication object including password and authentication string(?)
userData (optional): object, anything, e.g.: screenshot, comment, date/time

```
int getRankingPosition(gameID, playerID, orderBy='highscore', bool
higherIsBetter=true)
```

returns: Position of player in game, 1 is best
gameID: string to identify the game
playerID: string to identify the player
orderBy: string, which score to use for sorting if there are multiple scores, default is *'highscore'*
higherIsBetter: boolean, sorting order, the higher the score the better, default is *'true'*

```
RankedList getRankedList(gameID, rankStart, rankEnd, orderBy='highscore')
```

returns: an ordered array with entries of type RankedListEntry
gameID: string to identify the game
rankStart: int, the rank of the best player requested, for top ten this would be *'1'*
rankEnd: int, the rank of the worst player requested, for top ten this would be *'10'*
orderBy: string, which score to use for sorting if there are multiple scores, default is *'highscore'*

Notes for next version:

- Mark optional variables
- add 'higherIsBetter' for getRankedList
- include playerID in answer for getRankingPosition ??
- add option to only keep the highest score of a player
- add option for a maximum number of top entries to keep
- connect to Social Network SE
- use Identity Mangement GE for access control to creation and deletion of game tables

10.2 - Reality Mixer - Reflection Mapping SE

10.2.1 - Introduction to the Reflection Mapping SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. The Reality Mixer - Reflection Mapping SE is provided with the MIT License [60], see Terms and Conditions [63] for details.

10.2.1.1 - *Reflection Mapping SE Core*

- Provide a new level of realistic augmented reality rendering with the following steps.
- Pinpoint the centre and radius of the material sphere source relative to the Augmented Reality marker.
- Assign the camera image of the sphere to the source texturing of the target object(s)
- Perform spherical environment reflection mapping to the shading of the target object(s)
- Implementation via Xflow nodes in XML3D [18], cross platform Unity3D [10] script/package and reference algorithm pseudo-code

10.2.1.2 - *Intended Audience*

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Reality Mixer - Reflection Mapping SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the Reality Mixer - Reflection Mapping SE features. According to your usage, you should also be familiar with:

- XML3D [18]
- Xflow [64]
- Unity3D [10]

10.2.1.3 - *Provider of the Reflection Mapping SE*

For technical information please contact:

- The Walt Disney Company Ltd
- Professor Kenny Mitchell
- Tel.: +44 13 16 68 69 12
- Mail: kenny.mitchell@disneyresearch.com

or:

- DFKI GmbH – Agents and Simulated Reality
- Stefan Lemme
- Phone: +49 681 / 85775 - 5391
- Mail: stefan.lemme@dfki.de

For T&C information please contact:

- ETH Zurich, Switzerland
- Mr. Marcel Lancelle
- phone: +41 44 63 37 938
- email: marcel.lancelle(at)inf.ethz.ch

10.2.1.4 - *API Change History*

Revision Date	Changes Summary
Sep 12, 2013	initial version

10.2.1.5 - *How to Read This Document*

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., XML3D declaration or Xflow node operation.
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.2.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Pervasive Games Platform Specific Enablers [65].For more details about the usage of the Reflection Mapping SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2) and the architecture of the Pervasive Games Platform (see Section 5).

For more details about getting started with the Reality Mixer - Reflection Mapping SE within your own application or service, please refer to the Developer Guide [66] as a first starting point.

10.2.2 **- Overview of the Reflection Mapping SE**

Typically in Augmented Reality applications, objects appear lit without respect to real-world lighting surrounding the viewer. Here, surface normals (which hold surface orientation or shape) on the target virtual object are used to compute directional diffuse lighting or sometimes reflective specular highlights without concern for reflecting their true physical appearance. To provide a flexible solution for this we use the light falling upon a visible real sphere and map this to the virtual objects.

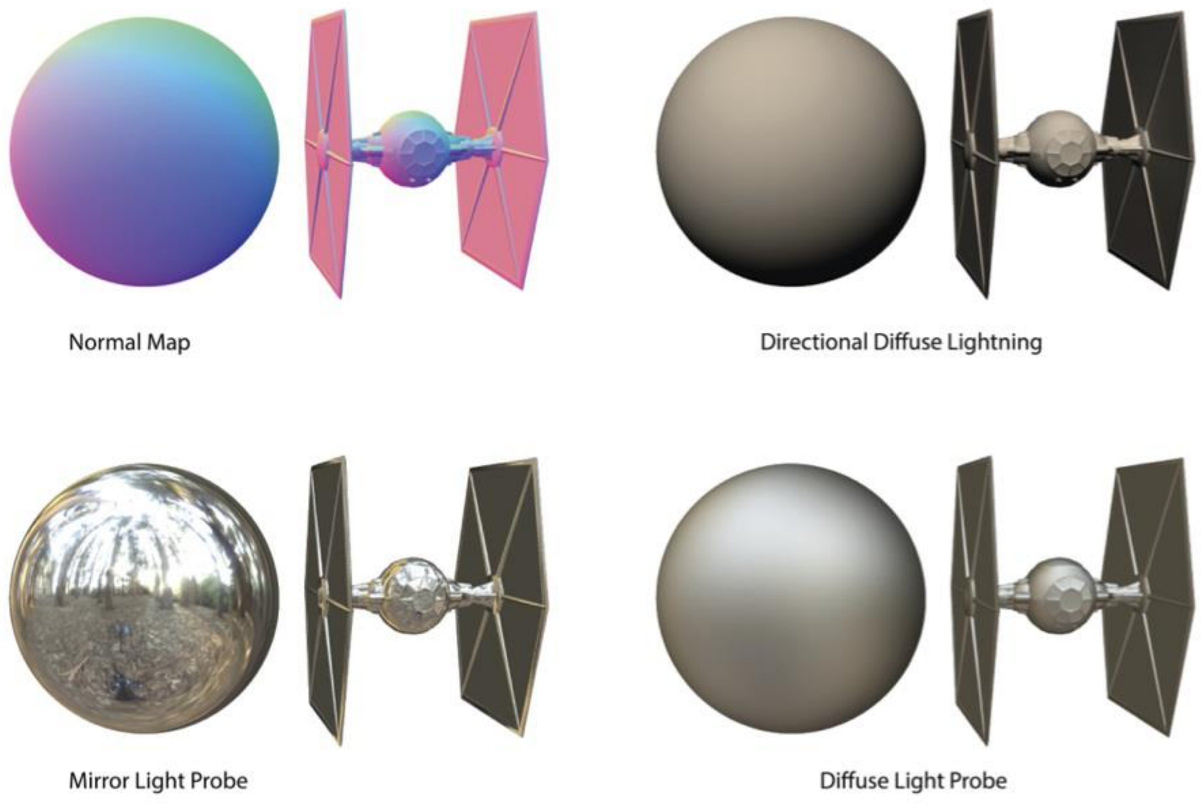


Figure 19 Reflection Mapping examples applied to a tie fighter model.

Model by SCORP [67] Environment map by Debevec [68]

If we use a chrome sphere, we get a shiny object appearance (bottom left). If we use a diffuse sphere (bottom right), we get a matt object appearance. Further, we can mix these effects and use interesting other material spheres to gain a variety of effects.

So in terms of implementation, this means there is a fixed spot next to an augmented reality marker that you always place the physical ball. When the camera points at the sphere, the reflection is captured into a texture. The bounds of this reflection texture are found by projecting the sphere's 3D position centre into screen space with a fixed scale offset along camera axis aligned to screen not greater than the radius of the ball. This then gives the texture coordinates within the camera's image. This circle is then looked up according to the normal of the vertices in the object, calculated with spherical environment mapping [69].

The XML3D/Xflow implementation of this Specific Enabler takes advantage of the Advanced-User Interface 3D-UI GE (see Section 6) and utilizes the hardware support of Xflow.

For native applications we provide a Unity3D [10] cross platform package.

10.2.3 - General Reflection Mapping SE API Information

In order to use the Reality Mixer - Reflection Mapping SE you need to import the respective JS libraries to and its dependencies, namely XML3D.

```
<head>
    <!-- ... //-->
    <script type="text/javascript" src="script/xml3d.js"></script>
    <script type="text/javascript" src="script/xflmix-reflection-
mapping.js"></script>
    <!-- ... //-->
</head>
```

Alternatively you simply need to get the Unity3D or jar package from ETH Zurich.

10.2.4 - Reflection Mapping SE API Specification

```
<data id="envmap"
    compute="spheremap = xflmix.lightprobe(arvideo, transform, lightprobe,
perspective)">

    <data src="#arBase"/>

    <float3 name="lightprobe">
        179.0  -5.0  50.0
    </float3>
</data>
```

The "data element" *envmap* utilizes the provided Xflow node *xflmix.lightprobe* to extract a spherical environment map using the light probe. Therefore, the position of the light probe is given as XML3D "float3" relative to the AR coordinate system. Moreover, the "data element" of the AR tracking technique, like the one of the Marker Tracking SE (see Section 10.5) is reused.

To apply the extracted spherical map to virtual object mimicking real-world lighting a new XML3D "shader" is introduced and applied to an XML3D "mesh". The *envlight* shader takes the spherical map as input and shades the given mesh based on this map.

```
<!-- Shader -->
<shader id="light-fx" script="urn:xml3d:shader:envlight">
    <data name="spheremap" src="#envmap"/>
</shader>

<!-- Group taking transformation from AR data and applying envlight shader -->
<group transform="#arbase" shader="#light-fx">
    <mesh type="triangles" src="{uri-to-mesh}" />
</group>
```

10.3 - Reality Mixer - Camera Artifact Rendering SE

10.3.1 - Introduction to the Camera Artifact Rendering SE

Different Terms and Conditions might apply for each of the FIcontent Enablers.The Reality Mixer - Camera Artifact Rendering SE is provided with the MIT License [60], see Terms and Conditions [70] for details.

10.3.1.1 - Camera Artifact Rendering SE Core

- This SE is supplied both as a Unity Package as well as a GLSL shader e.g. for the use in XML3D.
- Analyzes the camera video to create and add image artifacts to rendered virtual objects.
- In this release, only the motion blur effect will be available, others will follow in the next release.

10.3.1.2 - Intended Audience

This specification is intended for application developers. The document provides a specification of how to interoperate with the Augmented Reality Camera Artifact Renderer SE. To use this information, the reader should be familiar with:

- either Unity3D [10] development
- or XML3D [18]development

10.3.1.3 - Provider of the Camera Artifact Renderer SE

For technical information please contact:

- ETH Zurich, Switzerland
- Mr. Fabio Zünd
- phone: +41 44 63 20153
- email: fzuend(at)inf.ethz.ch

For Terms & Conditions information please contact:

- ETH Zurich, Switzerland
- Mr. Marcel Lancelle
- phone: +41 44 63 37 938
- email: marcel.lancelle(at)inf.ethz.ch

10.3.1.4 - API Change History

Revision Date	Changes Summary
Sep 12, 2013	first draft of this document

10.3.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., **void function(int a);**.
- An italic font is used to represent document titles or some other kind of special text, e.g., *STRING*.
- Variables are represented between brackets, e.g. {*id*} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.3.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Architecture of FIcontent Pervasive Games Platform Specific Enablers (see Section 5).

For more details about getting started with the Reality Mixer - Camera Artifact Rendering SE within your own application or service, please refer to the Installation Guide [71] as a first starting point if you are using Unity.

10.3.2 **- Overview of the Camera Artifact Rendering SE**

In see-through augmented reality applications, a camera, typically on the backside of a mobile device, captures video. At the same time the video is played live on the screen and virtual objects are rendered and overlaid in real-time. In general, the captured images and the rendered objects do not visually blend together well. For example, the camera sensor adds noise and the lens adds optical distortion to the captured images, but the rendered virtual objects do not exhibit any noise or distortion. However, these image effects, or artifacts, can be artificially created and applied. The artifacts can be sorted into the following categories:

- **External effects** happen due to external conditions and are not related to the camera parameters, such as motion blur or camera shake.
- **Lens filter effects** are caused by physical filters on the lens, like polarization filter, ND-filter, etc.
- **Lens effects** happen inside a lens due to physical properties and imperfections, such as flare, distortion, aberration, etc.
- **Shutter effects** appear since electronic or mechanical shutter can have different effects on the image (i.e. rolling shutter).
- **Sensor effects** depend on the sensor architecture type and result in noise, bloom, smearing, etc.
- **Post-processing effects** are usually applied after the images has been read from the sensor, like white balancing, tone mapping, compression, etc.

The Augmented Reality - Camera Artifact Rendering SE analyzes the captured video stream in order to calculate the appropriate effects and applies them to the rendered objects.

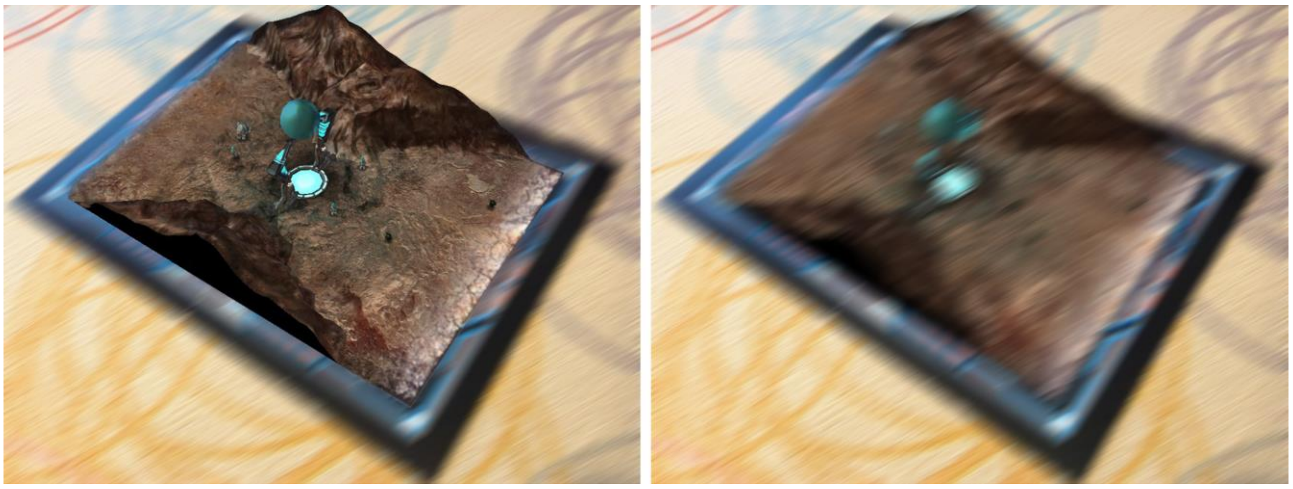


Figure 20 This mockup shows how motion blur helps to believably integrate virtual content.

Figure 1: This Camera Artifact Renderer mockup image depicts the effect of adding artificial camera blur to rendered objects. Left side: The camera moves and hence the background is blurred, but the scene from the game is not. Right side: The Camera Artifact Renderer interprets the image and adds appropriate blurring to the rendered scene to believably integrate the virtual content.

Please note that currently not all effects are implemented in the Augmented Reality Camera Artifact Renderer SE and some are work in progress.

10.3.3 **- General Camera Artifact Rendering SE API Information**

10.3.4 **- Camera Artifact Rendering SE API Specification**

10.3.4.1 - <u>Unity 3D integration</u>

The Reality Mixer Camera Artifact Renderer SE Unity package contains assets and scripts that can be loaded into a Unity project. Follow the Installation Guide [71] to integrate this SE into your Unity project.

Once the package is imported and set up, you see your virtual objects with added motion blur. To increase the amount of blur, increase the Blur Multiplier value in the CARLinearBlur component. To increase the temporal stability, increase the Stabilizer Strength value in the CARLinearBlur component. The SE may be enabled and disabled by enabling and disabling the CARLinearBlur component.

10.3.4.2 - <u>XML3D integration</u>

The implementation for XML3D (or other WebGL or OpenGL programs) is in development and will be released later.

10.4 - Augmented Reality - Fast Feature Tracking SE

10.4.1 **- Introduction to the Fast Feature Tracking SE**

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. The Fast Feature Tracking SE is provided with the MIT License [60], see Terms and Conditions [72] for details.

10.4.1.1 - <u>Fast Feature Tracking SE Core</u>

- Provide a markerless augmented reality tracking method (blob tracking) with the following steps.
- Pinpoint the centre and radius of the material sphere source relative to the Augmented Reality marker.
- Assign the camera image of the sphere to the source texturing of the target object(s)
- Perform spherical environment reflection mapping to the shading of the target object(s)
- Implementation via RESTful API, cross platform Unity3D [10] script/package and reference algorithm pseudo-code

10.4.1.2 - <u>Intended Audience</u>

This specification is intended for developers of augmented reality application. This document provides a specification of how to interoperate with the Augmented Reality - Fast Feature Tracking SE API. To use this information, the reader should firstly have a general understanding of Augmented Reality and feature tracking algorithms. According to your usage, you should also be familiar with Unity3D [10].

10.4.1.3 - <u>Provider of the Fast Feature Tracking SE</u>

For technical information please contact:

- The Walt Disney Company Ltd
- Professor Kenny Mitchell
- Tel.: +44 13 16 68 69 12
- Mail: kenny.mitchell@disneyresearch.com

For T&C information please contact:

- ETH Zurich, Switzerland
- Mr. Marcel Lancelle
- phone: +41 44 63 37 938
- email: marcel.lancelle(at)inf.ethz.ch

10.4.1.4 - <u>API Change History</u>

Revision Date	Changes Summary
Sep 12, 2013	initial version

10.4.1.5 - <u>How to Read This Document</u>

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., XML3D declaration or XFLOW node operation.
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.4.1.6 - <u>Additional Resources</u>

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Pervasive Games Platform Specific Enablers [65]. For more details about the usage of the Augmented Reality - Fast Feature Tracking SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2) and the architecture of the Pervasive Games Platform (see Section 5).

10.4.2 **- Overview of the Fast Feature Tracking SE**

Typically in Augmented Reality applications, one must use a pre-arranged QR-code or image marker to provide a frame of reference to the camera, so that virtual objects can be appear to be placed in the real world. Here we provide a solution where the application can learn the color of an object and track that color from then on.

The upcoming XML3D/Xflow implementation of this Specific Enabler takes advantage of the 3D User Interface (3D-UI) GE (see Section 6.2.3) and utilizes the hardware support of XFlow.

10.4.3 **- General Fast Feature Tracking SE API Information**

In order to use the Fast Feature Tracking SE you simply need to get the Unity3D or jar package from ETH Zurich. Alternatively you can get the free iPhone/iPad app via the AppStore [73].

10.4.3.1 - Data types

```
struct Color
{
   float r;
   float g;
   float b;
}
struct Position
{
   float x;
   float y;
   float z;
}
struct Video
{
   int width;
   int height;
   int framesPerSecond;
}
```

10.4.3.2 - Enumerations

```
enum FeatureTrackingMode
{
   kMeansCPU, MomentGPU
}
enum FeatureDebugViewMode
{
   Off, Gray, Distance, CentroidX, CentroidY, Centroid0X, Centroid0Y,
Centroid1X, Centroid1Y, KMeans
}
```

10.4.4 **- Fast Feature Tracking SE API Specification**

```
void arSetTrackingMode(FeatureTrackingMode trackMode)
```

trackMode: selects between CPU (kMeans [74]) and GPU (Moment of Integral [75]) tracking algorithms

```
void arCreateReductionResources(VideoInfo videoInfo)
void arReleaseReductionResources()
```

videoInfo: dimensions and desired frame rate of the camera video image

```
void arUpdateVideoFrame()
```

Provides cached access to the current video frame at videoRate refresh rate

```
void arProcessVideoFrame()
```

Performs GPU reduction operation on the video frame to calculate the center of the color distance matched blob feature. This step is separate to allow result caching and calculation at different rates.

```
bool arFoundFeature()
```

Returns whether the color was present at all in the image

```
void arCalculateCenter()
```

Performs the resulting CPU center calculation. This step is separate to allow result caching and calculation at different rates.

```
Position arGetFeatureCenter()
```

Returns the feature's center.

```
float arGetFeatureSize()
```

Returns the feature's size.

```
void arSetTrackingColor(Color col)
```

color: red, green, blur floating-point color components

```
void arEnableColorSampling(bool enable)
Color arGetSampledColor()
```

enable: enables/disables sampling of color region in the centre of the camera image, to identify the colored feature for subsequent tracking.

```
void arSetDebugViewMode(FeatureDebugViewMode debugMode)
```

debugMode: selects from a range of debugging views showing the fast feature tracking algorithm progress

10.5 - Augmented Reality - Marker Tracking SE

10.5.1 - Introduction to the Marker Tracking SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [76] attached.

10.5.1.1 - Marker Tracking SE Core

- HTML5
- JS library
- XML3D/Xflow integration
- performs AR marker detection on video streams

10.5.1.2 - Intended Audience

This specification is intended for developers of web applications interested in Augmented Reality applications on the web. This document provides a specification of how to interoperate with the Marker Tracking SE API. To use this information, the reader should firstly have a general understanding of Augmented Reality features. You should also be familiar with:

- HTML5
- JavaScript
- XML3D [18] and Xflow [64]
- usual image processing algorithms

10.5.1.3 - Provider of the Marker Tracking SE

- DFKI GmbH – Agents and Simulated Reality
- Stefan Lemme
- Phone: +49 681 / 85775 - 5391
- Mail: stefan.lemme@dfki.de

10.5.1.4 - API Change History

Revision Date	Changes Summary
Aug 9, 2013	initial version

10.5.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.5.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Pervasive Games Platform Specific Enablers [65].For more details about the usage of the Marker Tracking SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2) and the architecture of the Pervasive Games Platform (see Section 5).

10.5.2 - Overview of the Marker Tracking SE

The main functionality that the Marker SE provides is:

- AR marker detection and tracking on video streams
- Xflow node implementation
- XML3D integration

This Enabler provides Xflow node implementations to be used within the "data elements" of XML3D to receive the transformation information for given AR markers. Thereby, we take advantage of recent image processing algorithms.

10.5.3 - General Marker Tracking SE API Information

You need to import the respective JS libraries to utilize the Marker Tracking SE and its dependencies, namely XML3D.

```
<head>
    <!-- ... //-->
    <script type="text/javascript" src="script/xml3d.js"></script>
    <script type="text/javascript" src="script/xflar-marker-
tracking.js"></script>
    <!-- ... //-->
</head>
```

10.5.4 - Marker Tracking SE API Specification

The "data element" *arBase* utilizes the xflow node "xflar.detect" to provides the necessary information for the markers pose such as

- the *visibility* as bool,
- the transformation *transform* as XML3D 4x4 matrix,
- the *perspective* for the "view node" of a XML3D scene

The marker identifier is given as integer and will be interpreted as NyID of the fiducial marker.

```
<data id="arBase"
    compute="transform, visibility, perspective = xflar.detect(arvideo,
marker)">

    <int name="marker">{marker-id}</int>
    <data id="arvideoBase">
        <texture name="arvideo">
            <video src="{uri-of-videostream}" id="inputvideo"></video>
        </texture>
    </data>

</data>
```

The "video element" can be used to pass the webcam stream through the User Media API interface [77] of recent browsers. The output of this data element will be utilized within the actual XML3D scene.

```
<!-- Viewpoint with connection to AR data -->
<view id="View" perspective="#arBase" />

<group transform="#arBase" shader="#light-fx" >
    <mesh type="triangles" src="{uri-to-xml3d-mesh}" />
</group>
```

Please refer to the XML3D documentation [18] for further information on the utilization of transformations, meshes and cameras.

10.6 - Games with Things - Spatial Match Making SE

10.6.1 - Introduction to the Spatial Matchmaking SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [78] attached.

10.6.1.1 - Spatial Matchmaking SE Core

- Spatial Matchmaking Webservice - a Java Servlet that may be deployed to an Apache Tomcat server for example
- Unity package allowing Unity apps to use the service easily
- Unity Demo App illustrating the use of the Unity package

These elements are available from github: https:*github.com/gfoot/SpatialMatchmaking*

10.6.1.2 - Intended Audience

This specification is intended for server and application developers. This document provides a specification of how to interoperate using the Spatial Matchmaking SE API.

The reader should be familiar with:

- HTTP/1.1
- JSON data serialization

10.6.1.3 - Provider of the Spatial Matchmaking SE

For technical information please contact:

- Studio Gobo
- Mr James Callin
- Mail:info@studiogobo.com

10.6.1.4 - API Change History

^ Revision Date ^ Changes Summary

Sep 11, 2013	initial version
...	...

10.6.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.A **bold**, "monospaced" font is used to represent code or logical entities, e.g., HTTP method (**"GET, PUT, POST, DELETE"**).An *italic* font is used to represent document titles or some other kind of special text, e.g., URI.Variables are represented between brackets, e.g. *{id}* and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

10.6.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Pervasive Games Platform Specific Enablers [65].

For more details about getting started with the Spatial Matchmaking SE within your own application or service, please see the section below.

10.6.2 - Overview of the Spatial Matchmaking SE

The main functionality that the Spatial Matchmaking SE provides is:

- Allow a client to apply for pairwise matching with compatible clients based on client-supplied criteria such as geographic proximity requirements and key/value pair matches

Location data is supplied by the client in a widely-used format, allowing the client to use the Location GE or other means to determine its location.

The client communicates with the service through standard HTTP connections, which should be easily accessible from most environments. Some environments restrict the usage of HTTP (e.g. limiting choice of verbs, not allowing custom headers) but the service is designed to work even if only a subset of the HTTP protocol is available on the client.

Use a FMC diagram to explain the main components of this SE and the interaction with GEs, other SEs, and applications.

10.6.3 - General Spatial Matchmaking SE API Information

10.6.3.1 - Matchmaking Process

At a high level, clients go through three phases during the matchmaking process:

- Registration
- Querying for a match
- Connecting to other clients

If the final connection phase fails because the clients are not compatible - for example, network problems such as NATs prevent them from communicating with each other - then the clients can return to querying for other matches.

10.6.3.1.1 - Registration and client record updates

To register with the service, clients post some information about themselves, discussed in more detail in the API specification below. The server responds by directing the client to a specific URI representing that client.

The client can query its client record from the returned URI, and can also update the record at any later stage if the data changes - for example, location data can be update dynamically.

The client can also delete its record to cleanly terminate its involvement with the service.

10.6.3.1.2 - Querying for a match

Clients query for matches through a query URI. If a match is available then the server returns a redirect to a new URI containing details of the match. If no match is available then the server returns an empty response. In between, the server may hold the connection open for a period of time in the hope that a match will become available soon.

10.6.3.1.3 - Connecting to other clients

The connection protocol between clients is not specified here and is not a concern of the service. Clients may use any protocol they choose to form a connection. To facilitate, the client records include a "connectionInfo" string, whose format should be mutually understood by the clients.

When the clients are satisfied with their connection, they may delete their records from the server to indicate that they no longer require its services.

10.6.3.2 - Flow Diagrams

These diagrams may help to illustrate the process:

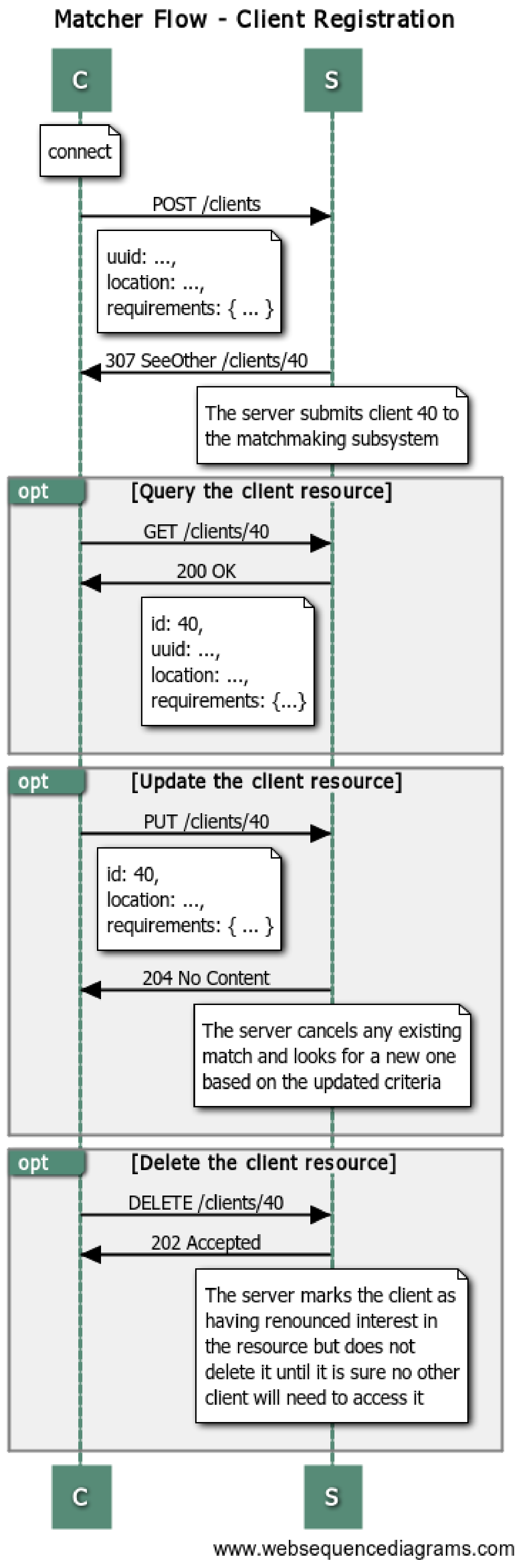

Matcher Flow - Client Registration
C
S
connect
POST /clients
uuid: ...,
location: ...,
requirements: { ... }
307 SeeOther /clients/40
The server submits client 40 to the matchmaking subsystem
opt
[Query the client resource]
GET /clients/40
200 OK
id: 40,
uuid: ...,
location: ...,
requirements: {...}
opt
[Update the client resource]
PUT /clients/40
id: 40,
location: ...,
requirements: { ... }
204 No Content
The server cancels any existing match and looks for a new one based on the updated criteria
opt
[Delete the client resource]
DELETE /clients/40
202 Accepted
The server marks the client as having renounced interest in the resource but does not delete it until it is sure no other client will need to access it
C
S
www.websequencediagrams.com

Figure 21 Matcher Flow - Client Registration

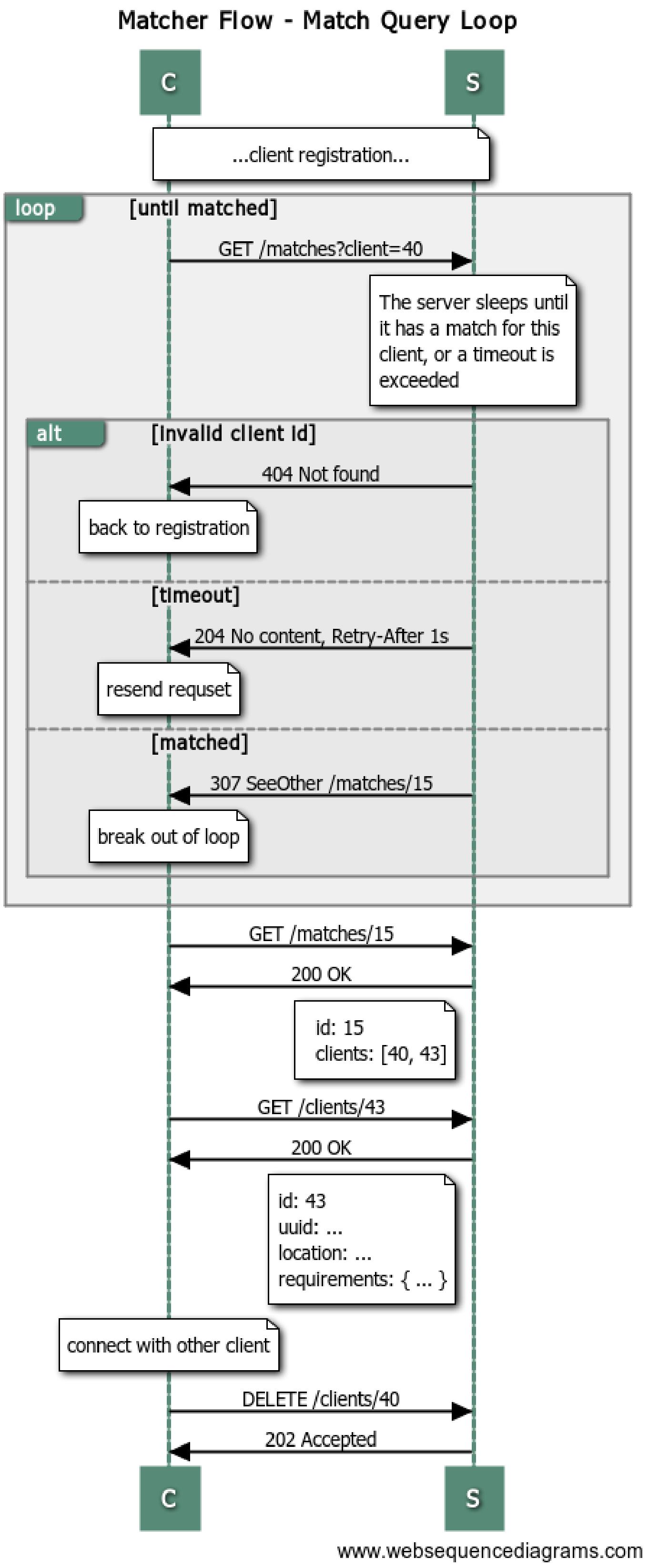


Figure 22 Matcher Flow - Match Query Loop

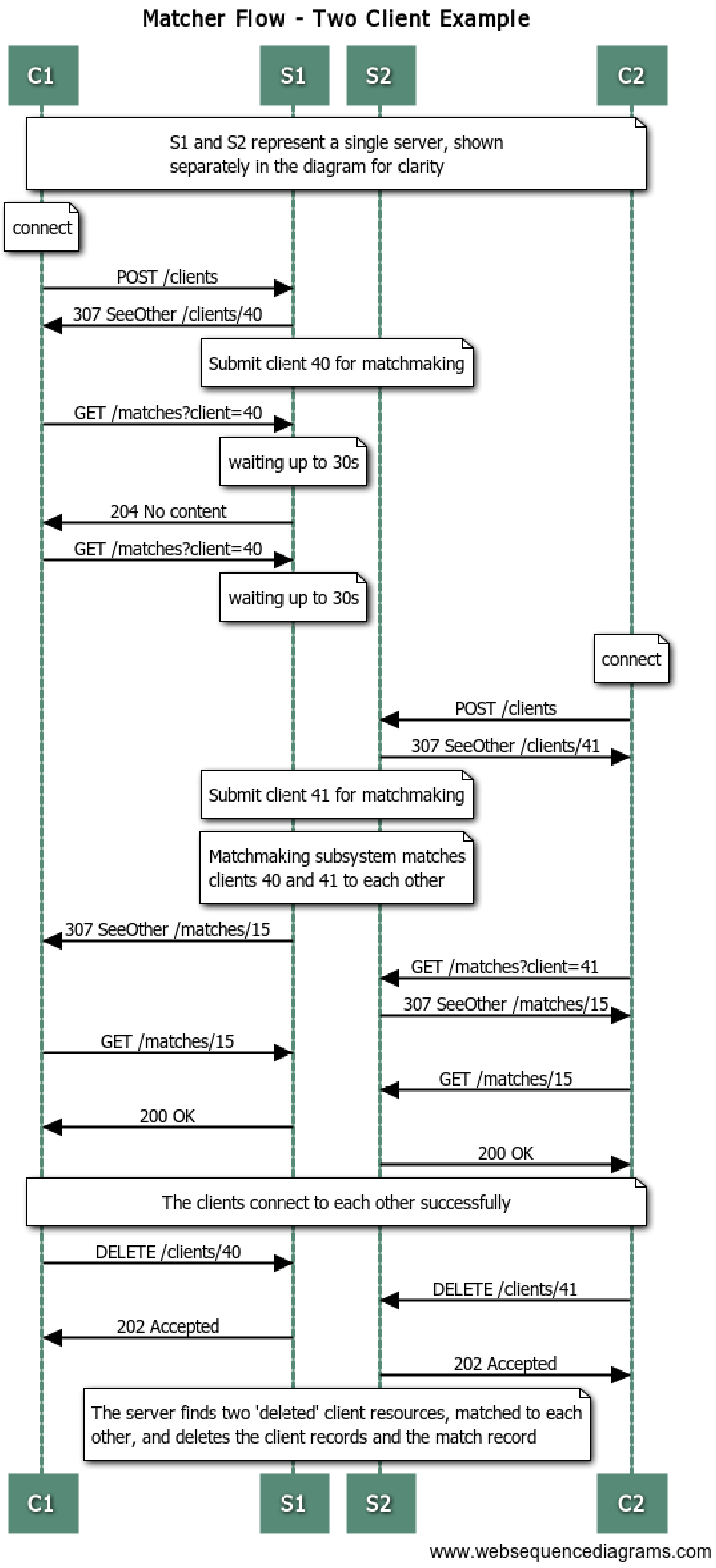

Matcher Flow - Two Client Example
C1
S1
S2
C2
S1 and S2 represent a single server, shown separately in the diagram for clarity
connect
POST /clients
307 SeeOther /clients/40
Submit client 40 for matchmaking
GET /matches?client=40
waiting up to 30s
204 No content
GET /matches?client=40
waiting up to 30s
connect
POST /clients
307 SeeOther /clients/41
Submit client 41 for matchmaking
Matchmaking subsystem matches clients 40 and 41 to each other
307 SeeOther /matches/15
GET /matches?client=41
307 SeeOther /matches/15
GET /matches/15
GET /matches/15
200 OK
200 OK
The clients connect to each other successfully
DELETE /clients/40
DELETE /clients/41
202 Accepted
202 Accepted
The server finds two 'deleted' client resources, matched to each other, and deletes the client records and the match record
C1
S1
S2
C2
www.websequencediagrams.com

Figure 23 Matcher Flow - Two Client Example

10.6.4 - Spatial Matchmaking SE API Specification

10.6.4.1.1 - /clients

Method	POST
Call	http:*[server]/clients*
Path Parameter	None
Query Parameter	None
Headers	Content-type: application/json
Body	The body must contain the item to ingest. It must be compliant to the Client Record Data Format below
Response	HTTP 201 with body of corresponding Client Record (see below)

10.6.4.1.2 - /clients/{cid}

Method	GET
Call	http:*[server]/clients/{cid}*
Path Parameter	**cid** Client Record id
Query Parameter	None
Headers	
Body	None
Response	HTTP 200 with body of corresponding Client Record (see below)
Method	PUT
Call	http:*[server]/clients/{cid}*
Path Parameter	**cid** Client Record id
Query Parameter	None
Headers	Content-type: application/json
Body	Client Record, as below. Accepts partial update, only fields present are altered.
Response	HTTP 204
Notes	The client should provide a partial or full client record in the message body. It is only necessary to include fields which are being updated with new values; omitted fields will retain their existing values. If the client is currently part of a match then the server must cancel the match if it is no longer suitable given the updated client record - for example, if

	new requirements invalidate the match, or if other attribute changes invalidate the match due to requirements specified by the other client in the match. The server should not cancel existing matches if the updated client record is still compatible with the other clients in the match. Notably this allows dynamic location updates without invalidating matches, and it allows clients to delay providing connectionInfo until after they are matched.
Method	DELETE
Call	http:*[server]/clients/{cid}*
Path Parameter	**cid** Client Record id
Query Parameter	None
Headers	None
Body	None
Response	HTTP 202
Notes	The server must not delete the client record if it is part of a match, unless all other client records in the match have also been marked for deletion, in which case the server should delete all the client records together, along with the match record.

10.6.4.1.3 - /clients/{cid}/update

Method	POST
Call	http:*[server]/clients/{cid}/update*
Path Parameter	**cid** Client Record id
Query Parameter	
Headers	Content-type: application/json
Body	Client Record, as below. Accepts partial update, only fields present are altered.
Response	
Notes	This is an alias for PUT on /clients/{cid}, and the client and server requirements are as specified above. It is provided purely to support poor quality HTTP client implementations which cannot use the PUT verb.

10.6.4.1.4 - /clients/{cid}/delete

Method	POST
Call	http:*[server]/clients/{cid}/delete*
Path Parameter	**cid** Client Record id
Query Parameter	None

Headers	
Body	None
Response	
Notes	This is an alias for DELETE on /clients/{cid}, and the client and server requirements are as specified above. It is provided purely to support poor quality HTTP client implementations which cannot use the DELETE verb.

==== /matches?client={cid} ====

Method	GET
Call	http:*[server]*matches?client={cid}
Path Parameter	none
Query Parameter	**cid** Client ID
Headers	Content-type: application/json
Body	None
Response	HTTP 303
Notes	Query and/or wait for a match for the specified client. If no match is available then the server may wait for an unspecified duration in anticipation of a match being found. If no match is found for longer than the server is prepared to wait, then the server should report success, but not provide any content, with a suggested Retry-After header. The client should not retry sooner than specified. Note that a server which does not support long-polling may require a longer retry duration to avoid high volumes of client requests. If the specified client ID does not correspond to a known client record the server should return a distinct error code so that clients know not to reissue the request.

10.6.4.1.5 - /matches/{mid}

Method	GET
Call	http:*[server]/matches/{mid}*
Path Parameter	**mid** Match ID
Query Parameter	None
Headers	Content-type: application/json
Body	None
Response	HTTP 200 with body Match Record
Notes	The server should respond with a match record specifying a consistently-ordered (but not necessarily sorted) list of the client IDs of the clients involved in the match. The order in which the clients are reported must be consistent in responses to all clients, and client software may rely on this consistency during the connection phase.

10.6.4.2 - Data formats

Where specified, message bodies may contain client records or match records. The sample implementation uses JSON formatting for all data, and client and server messages that contain message bodies should specify content type "application/json".

10.6.4.2.1 - Match record

Field	Type	Interpretation
id	integer	Match's ID
clients	List of client IDs	Consistently-ordered list of clients in this match

10.6.4.2.2 - Client record

Field	Type	Interpretation
id	integer	Client's ID
uuid	string	UUID supplied by the client
location	Location object	Client's reported location
requirements	List of requirement objects	Client's matchmaking requirements
connectionInfo	string	Client-defined information describing how to form a connection with this client

10.6.4.2.3 - Location object

Field	Type	Interpretation
latitude	double	Latitude in degrees, from -90 to +90
longitude	double	Longitude in degrees, from -180 to +180

10.6.4.2.4 - Requirement object

Field	Type	Interpretation
@type	string	Type of requirement

The specified requirement names are as follows, along with their additional fields and pass criteria:

10.6.4.2.4.1 - requireAttribute

Field	Type	Interpretation
attribute	string	Name of attribute
values	list of strings	List of acceptable values

This requirement passes for a group of clients if and only if both:

- all the clients specify an instance of this requirement with the same 'attribute' name
- at least one of the 'value' entries for this attribute is shared between all the clients in the prospective match

10.6.4.2.4.2 - requireLocationOverlap

Field	Type	Interpretation
radius	integer	Maximum travel distance in metres

This requirement passes for a group of clients if there is some geographic point which lies within all clients' specified radii. For purposes of this evaluation, if not all clients in the prospective match specify this requirement, but some do, then those that do not specify are treated as if they specified a radius of zero.

10.6.4.2.4.3 - *requireLocationWithin*

Field	Type	Interpretation
radius	integer	Maximum distance to other clients in metres

This requirement passes if all other clients' locations are within the specified radius of the client which specified this requirement.

10.6.4.2.4.4 - *requireNotUuid*

Field	Type	Interpretation
uuid	string	UUID of client to ignore

This requirement passes for matches which do not include the specific excluded client. Primarily this allows clients to recover from connection failures, ensuring they are not matched against each other in future.

10.7 - Game Synchronization SE

10.7.1 - Introduction to the Game Synchronization SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [79] attached.

10.7.1.1 - <u>Game Synchronization SE Core</u>

- *type of this SE: C# library*
- *Synchronizes the game content among players*
- *Provides different game networking models*

10.7.1.2 - <u>Intended Audience</u>

This specification is intended for application developers. The document provides a specification of how to interoperate with the Game Synchronization SE API. To use this information, the reader should firstly have a general understanding of Network programming. You should also be familiar with:

- Networking protocols (IP, UDP, TCP).
- Socket Programming.
- Peer-to-Peer vs Server-to-Client.

10.7.1.3 - <u>Provider of the Game Synchronization SE</u>

- Marcel Lancelle, ETHZ
- Chino Noris, DRZ

10.7.1.4 - <u>API Change History</u>

Revision Date	Changes Summary
Sep 13, 2013	Version APIv0.1 drafted

10.7.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- An italic font is used to represent document titles or some other kind of special text, e.g., URI.
- A bold font is used to represent class names and interfaces.
- A bold and italic font is used to represent methods, properties and attributes.

10.7.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Gaming Specific Enablers [65].For more details about the usage of the Game Synchronization SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2), in specific the architecture of the Pervasive Games Platform (see Section 5).

For more details about getting started with the Game Synchronization SE within your own application or service, please refer to the Developer's Guide [80] as a first starting point.

10.7.2 **- Overview of the Game Synchronization SE**

The main functionality that the Game Synchronization SE provides is:

- RTS Lockstep mechanism for peer-to-peer synchronization of player's actions throughout the game.
- Authoritarian client (host) for peer-to-peer synchronization of the game state throughout the game.
- Authoritarian server for server-to-client synchronization of the game state throughout the game.

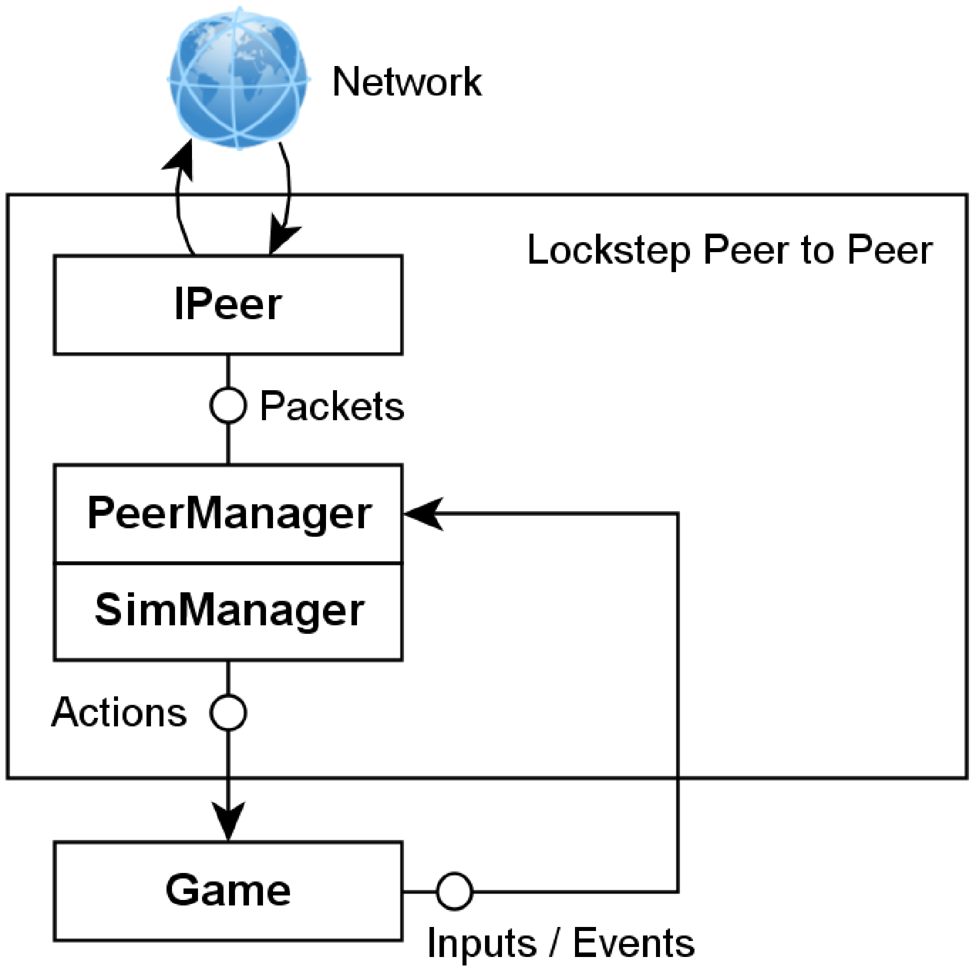


Figure 24 The RTS Lockstep mechanism of the Game Synchronization SE

The diagram above shows the RTS Lockstep mechanism

10.7.3 - Game Synchronization SE API Specification

10.7.3.1 - RTS-LockStep Model

Disclaimer: A crucial requirement of the RTS-Lockstep model is that the your game simulation must be deterministic, i.e. the same inputs will always produce the same output on all devices you plan to support. If this requirement is not met, the simulations will diverge, leading to different states of the game on different clients. Recovering from such de-synchronization is often complicated. If your game cannot be simulated deterministically, consider changing to a different networking model.

In the RTS-Lockstep model, the concept is that the clients participating to a game session only communicate the actions of the players, and then locally simulate the effect of those. Each peer participating to the game is identified by a unique integer called ***pid***. The data sent between peers consist of actions associated with a ***snap***, an integer counting the simulation steps from the start of the game. When a player gives an input on a client, the client registers this action and schedules it to be executed in the near future. All clients then share the actions that are about to happen, and when a client has all the actions of all players associated with a particular snap, it executes a simulation step.

The following sections include more details about this mechanism and highlights the main actors.

10.7.3.2 - Reliable delivery of non-reliable packets

When a *UDP* socket connection is used, an acknowledgement mechanism is provided, where a client sends to its peer its own actions, as well as the snap of the last action it received from them. This informs those peers what information is missing on the client.

The class responsible for this is **SnapAction**, which has two main purposes. First, it acts as Dictionary to associate the actions the client over time (a history of all local commands). Second, it stores the last set of actions received by each peer.

10.7.3.3 - Peer-To-Peer Connection

```
class PeerManager{
    internal SnapAction actions;
    public void Update();
}
```

The reliable distribution of packets is then provided by the **PeerManager** class. This class holds the association of the client ***pid*** with their network addresses and has the ability to send them data.

The ***Update*** method has to be called at each network cycle. For each peer, the **PeerManager** instance will send all the relevant recent actions, from the last received one (stored in the ***actions*** object) until the current snap.

10.7.3.4 - Simulation

```
class SimManager {
    public uint simSnap; // current simulation snapshot
    public bool ExecuteActions();
}
```

The **SimManager** class is in charge of simulating the effect that players actions have on the game. The ***simSnap*** variable holds the value of the current snap, i.e. the current time of the game. When the actions of all players associated to the current snap are available, the ***ExecuteAction*** method is called, allowing the **SimManager** to move the simulation forward by one snap.

10.7.3.5 - Players Actions

```
public enum ActionType : int
{
    Create,
    Move,
    Attack
}

class PlayerActionManager{
    internal void Create(UnitType unit, Vector3 position); // spawns a unit at a
location
    internal void Move(int objID, Vector3 destination); // orders to a unit to
move to a location
    internal void Attack(int objID, int targetID); // orders to a unit to attack
a target
}
```

The class **PlayerActionManager** is responsible to convert the player's input into actions. A set of functions is provided for common actions. This class has to be extended to include the specific actions a game may need.

```
class PeerManager {
    public uint snapActionDelay = 2;
    public virtual void AddAction(AbstractAction action);
}
```

The creation of an action within the **PlayerActionManager**, automatically triggers a call to the ***AddAction*** method in **PeerManager**. In order to compensate for the time it takes for clients to communicate, a delay is added to when the action is actually executed in game*. This delay is defined by ***snapActionDelay*** and is measured in number of snaps. This value can be converted into time by considering the simulation rate of your game, and should be set according to the expected latency between the peers.

For instance, if your game runs the simulation every 50ms (20 FPS sim), and the maximum expected latency between the peers is 150ms, ***snapsActionDelay*** should be set to 3. A higher number will add extra delay and eventually result in the game feeling less responsive, while a lower number may cause the game to temporarily stop on a client that is missing information from one of its peers.

*Notice that this is usually decoupled from the UI response to the user, which can right away give the illusion of a prompt execution (for example playing a "Yes, sir!" sound), even if the actual action will be executed later.

10.7.3.6 - Packets and Sockets

Each packet contains a list of actions of a player for a specific snapshot.The list can be empty if the player or his units didn't do anything during that snapshot. The packet and its actions both implement the interface **System.Runtime.Serialization.ISerializable**. It is possible to provide your own serialization methods by implementing the **IPacketSerializer** interface.

The network layer is provided by the **IPeer** interface and partially implemented in the **AbstractPeer** class. In the current implementation the **UDPPeer** class is provided with basic primitives to send and receive via *UDP*.

10.7.3.7 - Checksum and De-Synchronization

If the game simulation is fully deterministic, the RTS-Lockstep guarantees a synchronized game state across all clients. In case of doubt, the interface **ICheckSum** and the class **ICheckSumParam** provide support for checking if the game state is the same on all clients. This is done by defining a number of game-state properties one wants to check (like the position of all units), and then computing a unique hash tag, which can be compared to the one of the other peers.

In case a mismatch is detected, it's up to the game developer to choose what strategy to adopt. The first problem is to establish an authority who can rule on the actual game state. This may not be possible if only two clients are present. When more peers are present, hopefully two or more will agree, simplifying the choice.The second problem is how to reconstruct the proper game-state in the de-synchronized clients. If the game-state is relatively small, it can be transferred from the other clients. If the game state is too large, a client could re-simulate locally the match up to the de-synchronization. Both solutions may not be practical depending on your case, and often developers prefer to deal with the problem of making the simulation fully deterministic, rather dealing with this scenario.

11 - Common Specific Enablers of FIcontent platforms

In the following section we describe the Specific Enablers utilized by certain platforms. Thus, they were reclassified as common Specific Enablers. This particularly includes Enablers which provide a generic rather than domain-specific functionality, such as the Social Network SE, and thus will be used by at least two of the three platforms.

11.1 - Social Network SE

11.1.1 - Introduction to the Social Network SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check for each FIcontent Specific Enabler the Terms and Conditions [81] attached.

11.1.1.1 - Social Network SE Core

The Social Network SE Core (or SNE) is a REST Service with a Web interface that gives end users the opportunity to communicate with each other. Unlike monolithic infrastructures (like Facebook) the SNE provides not only full autonomy of the user data but also gives the opportunity to run it as a federated service.

The service includes

- account creation and deletion
- profile editing and password changes
- posting of status updates (text, images, etc.)
- 'following' of users
- commenting and 'liking' of posts

11.1.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Social Network SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the social network features. You should also be familiar with:

- RESTful web services
- HTTP/1.1
- OAuth 1.0

11.1.1.3 - Provider of the Social Network SE

- Dirk Krause, Pixelpark
- Cäcilienkloster 2, D-50676 Köln, Tel +49.221.951515-72

11.1.1.4 - API Change History

Revision Date	Changes Summary
Aug 22, 2013	initial version

11.1.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., HTTP method (GET, PUT, POST, DELETE).
- Variables are represented between brackets, e.g. {id} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

11.1.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Common Specific Enablers [82].For more details about the usage of the Social Network SE within the FIcontent platforms, please refer to the FIcontent Architecture (see Section 2), the Smart City Platform Architecture (see Section 4), and the Pervasive Games Platform Architecture (see Section 5).

For more details about getting started with the Social Network SE within your own application or service, please refer to the Developer Guide [83] as a first starting point.

***11.1.2* - Overview of the Social Network SE**

The main functionality that the Social Network SE provides is:

- User authentication (vs. GE)
- User participation and collaboration
- User generated content

The Social Network SE is a combination of services plugged together to provide above functionalities. As such it is a construction of open source software solutions with their respective documentation found elsewhere. This is especially true for the specification below which is mostly a copy of the pump.io documentation.It provides a *type of API* interface to the user.

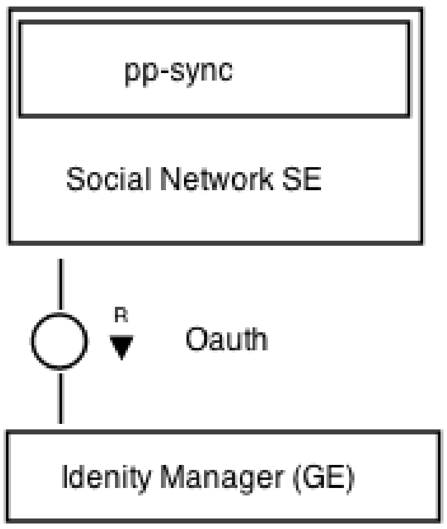


Figure 25 Connection of the Social Network SE to the Identity Management GE

It is planned to couple this service with the POI services of the Smart City Platform (see Section 4).

11.1.3 - General Social Network SE API Information

11.1.4 - Social Network SE API Specification

11.1.4.1 - Access databases

11.1.4.1.1 - URL scheme of API requests

The basic URL structure consists of the hostname (incl. port) and the name of the database you want to access:

```
http://localhost:5984/databasename
```

11.1.4.1.2 - Operations

To get a list of all databases on your server, perform a GET request at

```
http://localhost:5984/_all_dbs
```

11.1.4.1.2.1 - PUT

To create a new database, perform a PUT operation at

```
http://localhost:5984/mynewdatabase/
```

This will return a HTTP status of 201 if it was created or 412 if an error occured.

11.1.4.1.2.2 - DELETE

To delete a database, perform a DELETE operation at

```
http://localhost:5984/mynewdatabase/
```

This will return a HTTP status of 200 if it was deleted or 404 if the database does not exist.

For detailed description, refer to the CouchDB Wiki [84]

11.1.4.2 - Access documents

11.1.4.2.1 - URL scheme of API requests

The basic URL structure consists of the hostname (incl. port), the name of the database and the document id:

```
http://localhost:5984/databasename/doc_id_E4A897-B8D49
```

11.1.4.2.2 - Document structure

A document is represented as JSON object. A simple structure looks like this:

```
{
     "_id":"doc_id_E4A897-B8D49",
     "_rev":"D1C946B7",
     "msg": "SNE is really cool!",
     "user":{
          "id": "123",
          "name": "joe"
     },
     "type": "POST"
}
```

11.1.4.2.3 - Special fields

Note that there are some fields starting with a "_" prefix which are reserved for internal usage. Besides "_attachments", "_deleted", "_revisions", "_revs_info", "_conflicts", "_deleted_conflicts" and "_local_seq" there are the special mandatory and immutable fields:

- "_id": represents the unique identifier of the document
- "_revs": represents the current revision of the document

When a document is created, it is possible to set the id manually (PUT operation) or let the database setup the value for you (POST operation).

11.1.4.2.4 - Operations

To get a list of all documents in a database, perform a GET request at

```
http://localhost:5984/databasename/_all_docs
```

11.1.4.2.4.1 - GET

To retrieve a document, perform a GET operation at

```
http://localhost:5984/databasename/doc_id_E4A897-B8D49
```

This will return a HTTP status of 200 if it was successfully retrieved or 404 if it was not found.

11.1.4.2.4.2 - PUT

To create a new named document (means, that you set the id manually in your request), perform a PUT operation at

```
http://localhost:5984/databasename/new_doc_id
```

with Content-Type: "application/json"

and data like

```
{
     "msg": "SNE is really cool!",
     "user":{
          "id": "123",
          "name": "joe"
     },
     "type": "POST"
}
```

This will return a HTTP status of 201 if the document was successfully created or 409 if a conflict error occurred.The operation will set up the value "new_doc_id" for the document's field "_id".

11.1.4.2.4.3 - POST

To create a new document without naming, perform a POST operation at the database's URL:

```
http://localhost:5984/databasename/
```

with Content-Type: "application/json"

and data like

```
{
     "msg": "SNE is really cool!",
     "user":{
          "id": "123",
          "name": "joe"
     },
     "type": "POST"
}
```

This will return a HTTP status of 201 if the document was successfully created.The operation will generate a new document id automatically.

11.1.4.2.4.4 - DELETE

To delete a document, perform a DELETE operation at

```
http://localhost:5984/databasename/doc_id?rev=D1C946B7
```

This will return a HTTP status of 200 if the document was successfully deleted.

For detailed description, refer to the CouchDB Wiki [85]

11.2 - Content Sharing SE

11.2.1 - Introduction to the Content Sharing SE

Different Terms and Conditions might apply for each of the FIcontent Enablers.The Content Sharing SE is provided with the "licences", see Terms and Conditions [86] for details.

11.2.1.1 - Content Sharing SE Core

- Two pieces of software:
- An Android Library, to be used during development of mobile applications,
- A web service, exposing a REST interface to manage content.

11.2.1.2 - Intended Audience

This specification is intended for application developers. This document provides a specification of how to interoperate with Content Sharing SE interface. The reader should be familiar with:

- Android software development
- RESTful web services

11.2.1.3 - Provider of the Content Sharing SE

For technical information, please contact:

- Thales Communications & Security, Gennevilliers, France
- Mr. Farid BENBADIS
- phone: +33 1 41 303 929
- email: farid.benbadis+content(at)gmail.com

11.2.1.4 - API Change History

Revision Date	Changes Summary
Sep 16, 2013	fix APIv0.9 for FIcontent platform release 09/13

11.2.1.5 - How to Read This Document

Some special notations are applied to differentiate some special words or concepts. The following list summarizes these special notations.

- A bold, mono-spaced font is used to represent code or logical entities, e.g., **void function(int a);**.
- An italic font is used to represent document titles or some other kind of special text, e.g., *STRING*.
- Variables are represented between brackets, e.g. {*id*} and in italic font. When the reader finds one, it can assume that the variable can be changed by any value.

11.2.1.6 - Additional Resources

You can download the most current version of this document from the FIcontent platform documentation website at the Roadmap of FIcontent Common Specific Enablers [82].

For more details about getting started with the Content Sharing SE within your own application or service, please refer to the Developer Guide [87] as a first starting point.

***11.2.2* - Overview of the Content Sharing SE**

The main functionality that the Content Sharing SE provides is:

- Transparent content synchronization with regards to underlying network connectivity (infra, ad hoc)
- Synchronization of feeds containing linked content (comments on images, images related to one another…)

The Content Sharing SE is a combination of services plugged together into an Android Library to provide above functionalities.The Content Sharing SE uses the Identity Management GE to authenticate a user when subscribing a feed and/or submitting an entry. The requests are direct access to the database.

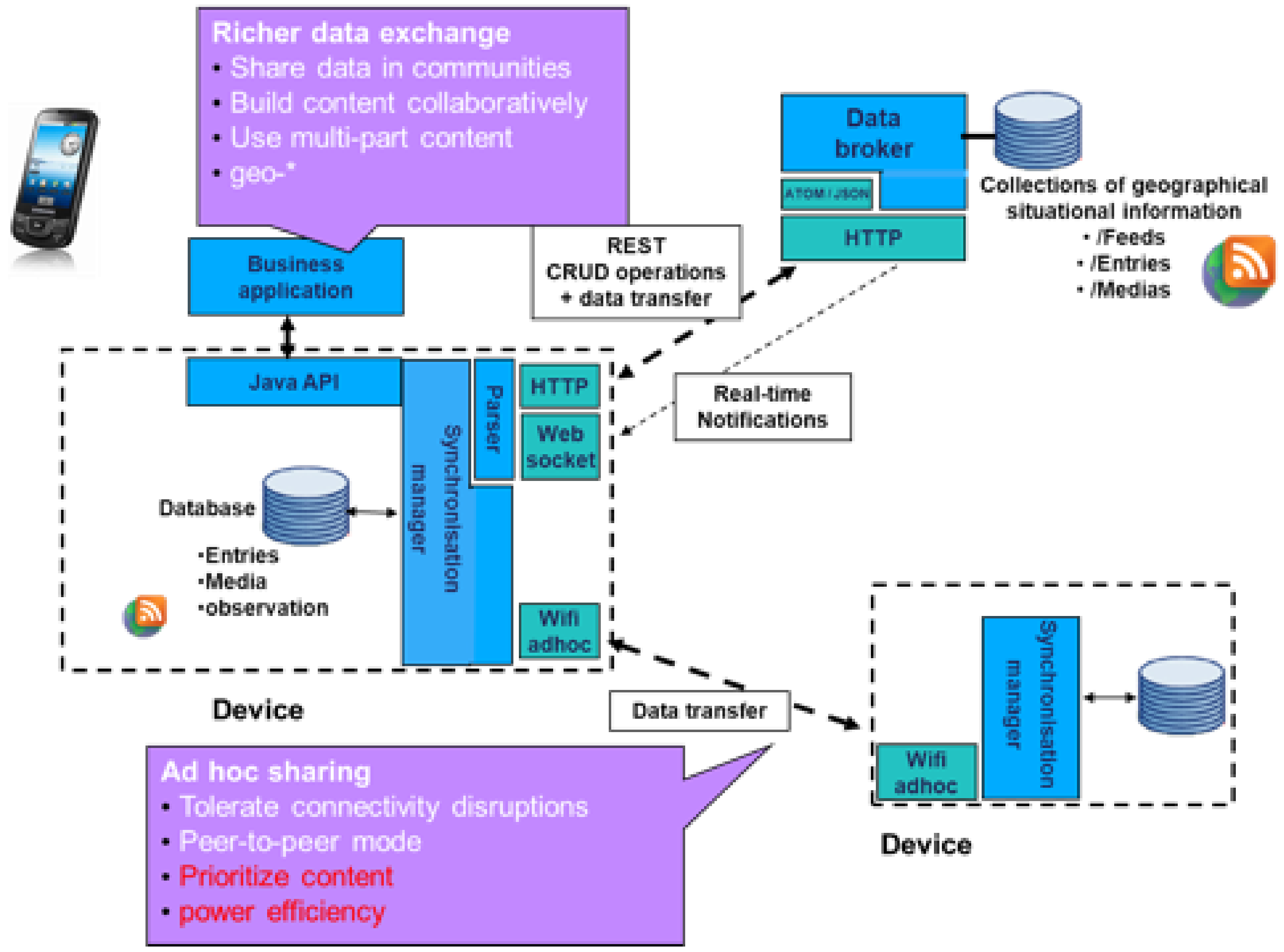


Figure 26 Overall structure of the Content Sharing SE library

11.2.3 - General Content Sharing SE API Information

In the infrastructure part of the architecture, we provide ability to establish communications between users and an Atom/JSON server.The exchanges between the server and the users are based on the Atom Publishing Protocol (APP). The server delivers an Atom feed, composed of a set of entries. The data received though the Atom feed is stored in a user local database. To synchronize data with the server, it delivers an Atom feed. This stream called "Feed" is composed of entries called "Entry". We collect and analyze the flow to get the data. These data will be recorded and will fed a database located on the terminal.The server is built on top of the Apache Abdera project [88] which aim is to build a functionally-complete, high-performance implementation of the Atom Syndication Format and Atom Publishing Protocol specifications.

The server stores content following an RSS representation of XML as depicted by the figure below. The Atom Syndication Format is an open document format based on XML designed for syndicating periodical content such as blogs or news sites. As described in Figure 1, nodes subscribe to a feed and upload the entries in this feed. Each entry can contain any type of content.

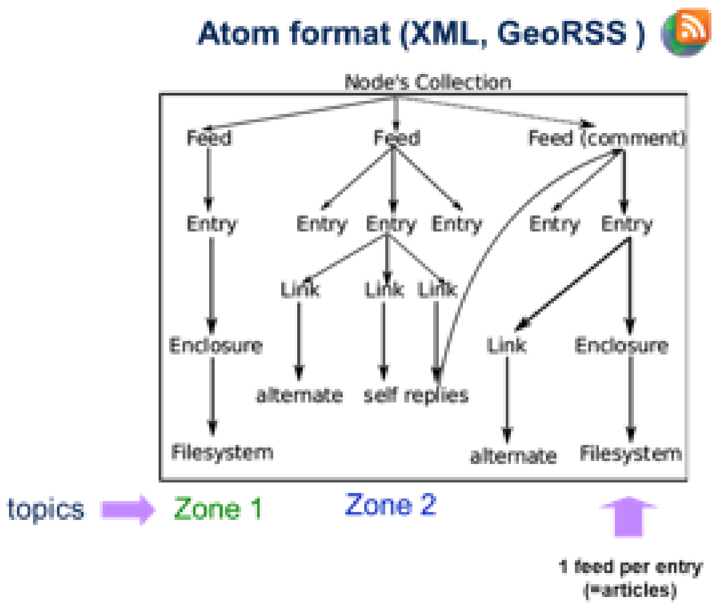


Figure 27 Structure of the Atom Syndication Format

Data type: Entry:

```
POST /entries HTTP/1.1 Host: myaddress
Content-Type: application/atom+xml;
charset=utf-8
Content-Length: nnn
<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns="http://www.w3.org/2005/Atom">
     <author><name>Thales</name></author>
     <link href="/entries/1185665" rel="edit" type="application/atom+xml"
length="0" />
     <id>1185665</id>
     <title type="html"></title>
     <summary type="html">...short summary…</summary>
     <updated>2013-07-09T08:19:42.959Z</updated>
     <content type="html">
          … here it is the object itself …
     </content>
</entry>
HTTP/1.1 201 Created
```

The Diagram and the data type above shows the representation of a message and its related entries.

11.2.4 - Content Sharing SE API Specification

11.2.4.1.1 - RESTful API Functions:

All data is encoded in Atom. For details on parameters and returned objects please refer to the SFS2X description below.

description	HTTP method	URL	parameters
submit entry	POST	/FeedID/Entries/ID	entries, authentication, userData
submit comment	POST	/FeedID/EntryID/comment-id	entries, authentication, userData

get entry or comment	GET	/FeedID/EntryID/ID	

11.2.4.1.2 - Android Library call API Functions:

Class FeedSyncInfra*This class lets the user to synchronize some feeds, some entries from a server in infrastruture mode.*

`Cursor getCursorEntries()`

returns: The cursor of the database to access to entries

`Cursor getCursorFeeds()`

returns: The cursor of the database to access to feeds

`List<Entry> getStringListDB()`

returns: The list of entries of the database in chronological order

`List<Entry> getListEntry()`

returns: The list of entries of the database in chronological order

`int getIdEntry(int positionExtras)`

returns: The id of the entry as a function of the position(e.g. in a listview)

`int getIdEntry(int positionExtras)`

returns: The id of the entry as a function of the position(e.g. in a listview)

`void LaunchLoadingFeeds()`

Launch the asynctask for the loading of the feeds

`void syncAllCommentsWithServer()`

This function synchronized all comments to the server

11.3 - Content Enrichment SE

11.3.1 - Introduction to the Content Enrichment SE

The FIcontent Specific Enablers are owned by different partners. Therefore, different Terms and Conditions might apply. Please check the Terms and Conditions [89] for the Content Enrichment SE.

11.3.1.1 - Content Enrichment SE Core

- REST webservice, JSON data serialization formats
- Add videos to POIs
- Tagging a video (create and display a video with content enrichment)
- Give feedback about a POI (comments, rating)

11.3.1.2 - Intended Audience

This specification is intended for both application developers and service providers. For the former, this document provides a specification of how to interoperate with the Content Enrichment SE API. For the latter, this specification indicates the interface to be provided in order for clients to interoperate with services which benefit from the provided functionality. To use this information, the reader should firstly have a general understanding of the Content Enrichment features. You should also be familiar with:

- RESTful web services
- HTTP/1.1

- JSON data serialization formats

11.3.1.3 - Provider of the Content Enrichment SE

- Fraunhofer Institute for Open Communication Systems FOKUS
- email: robert.seeliger@fokus.fraunhofer.de

11.3.1.4 - API Change History

Revision Date	Changes Summary
Aug 22, 2013	initial version
Feb 27, 2014	updated version
Apr 16, 2014	updated version

11.3.2 - Overview of the Content Enrichment SE

Content Enrichment provides functions to create, distribute and play interactive video content across platforms and devices by making object in the video clickable for the viewer. It provides interfaces to incorporate web 2.0 capabilities and community functionalities as well. Thus the enabler acts as a common building block in future video and multimedia infrastructures, to allow seamless, platform independent and convenient enrichment of any type of video content using any type of device for a plurality of application cases covering UGC, professional content, advertisement as well as edutainment.

The Content Enrichment SE provides the following capabilities:

- Main component for all enrichment relevant work items
- Object linkage to supplemental / related information
- Social Media connector
- Connection of external resources (metadata bases, external tools for image recognition etc.)

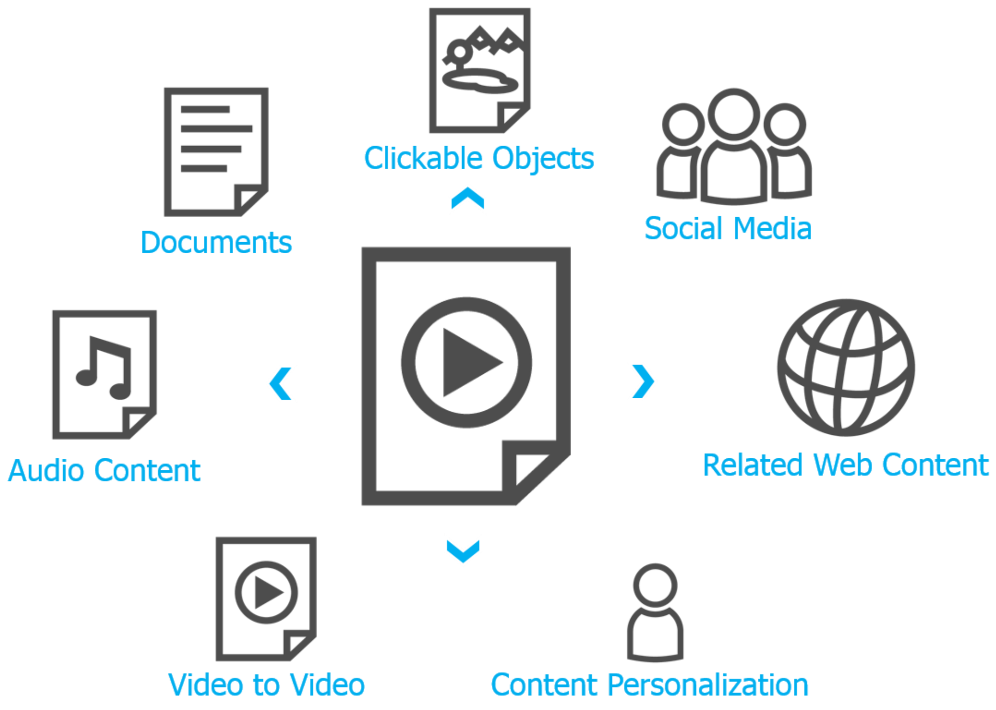


Figure 28 Overview of the Content Enrichment SE

Content Enrichment SE is based on Fraunhofer FOKUS background technology non-linear-video (NLV). It can be used by all project partners during the term of the PPP projects for testing and experimentation. For any other use (e.g. commercial use) of the Content Enrichment SE, please contact us.

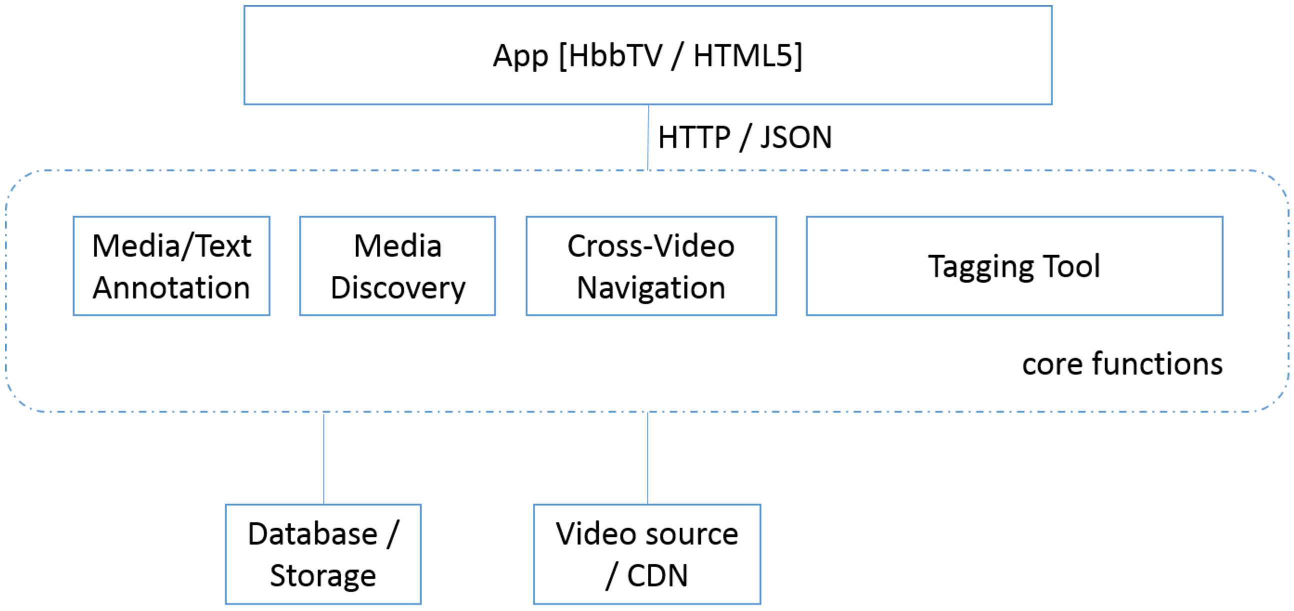


Figure 29 Architecture of the Content Enrichment SE

11.3.3 - General Content Enrichment SE API Information

Please refer to the Content Enrichment SE API Information [90].

11.3.4 - Content Enrichment SE API Specification

In general there are always the same types of functions available:

SAVE, ADD, UPDATE, DELETE, GET and **GET-LIST**

- **SAVE** is a shortcut for Update OR Add and uses the presence of numeric "ID" fields to determine the appropriate action. it is currently only used for "Config"
- **ADD** and UPDATE take the current object via $data in JSON format. UPDATE requires an ID for the top object but not for lower level objects (if $recursive is used)
- **GET** returns the requested object
- **GET-LIST** returns a list of objects for the given parent object
- **$data** consists of the adressed object in JSON form
- only **ONE** object is contained in **$data** but it can contain multiple objects of sub-categories (e.g. **addAppearance** can have 1 or more KeyFrames attached if *$recursive=true* is used)
- **$Param** is special for Config but is essentially equal to **$ID** elsewhere because each parameter can only be there once per video

<u>Function list</u>

```
saveAppearance ( $data, [$recursive = false] )
addAppearance ( $data, [$recursive = false] )
updateAppearance ( $data, [$recursive = false] )
deleteAppearance ( $ID )
getAppearance ( $ID )
getAppearanceList ( $Object_ID )

saveConfig ( $data )
deleteConfig ( $Param, $Video_ID )
getConfig ( $Param, $Video_ID )
getConfigList ( $Video_ID )

saveObject ( $data, [$recursive = false] )
addObject ( $data, [$recursive = false] )
updateObject ( $data, [$recursive = false] )
updateDoLink ( $ID, $DirectOpen )  // special: updates "direct open link", now
known as "quick link". requires the TYPE of the link or an empty string to reset
it
deleteObject ( $ID )
getObject ( $ID )
getObjectList ( $Video_ID )

saveLink ( $data )
addLink ( $data )
updateLink ( $data )
deleteLink ( $ID )
getLink ( $ID )
getLinkList ( $Object_ID )

copyVideo ( $ID, [$API_to = false] )  // special: allows to copy a video from
one api to the next. current api is assumed as target if no second parameter is
provided
copyVideoUID ( $UID, [$API_to = false] ) // special: same as copyVideo() but
with UID instead of ID

saveVideo ( $data, [$recursive = false] )
addVideo ( $data, [$recursive = false] )
updatePublic ( $ID, $Public )   // special: requires the Video-ID and the
Public-state (0/1)
updateVideo ( $data, [$recursive = false] )
deleteVideo ( $ID )
getVideo ( $ID, $recursive )
getVideoByUID ( $UID, [$recursive = true] ) // special: same as getVideo but for
UIDs
getVideoList ( [$recursive = false] )
getVideos ( [$extended = false] )  // special: only returns UID, Name and Public
(and ID and Duration in extended mode)

savePoint ( $data )
addPoints ( $ID, $dataArr )            // special: used to add a point to all
keyframes of the current appearace at once, reducing round-trips.
```

```
                              // $ID is the array-index of the point, $dataArr is
and ARRAY of {"Shape_ID":keyFrame.ID, "X":keyFrame.points[$ID]["X"],
"Y":keyFrame.points[$ID]["Y"]}
                              // --> client must find all keyframes for the
current appearance or this will cause errors!
updatePoint ( $data )
deletePoints ( $ID, $KeyFrame_IDs )  // special: deletes a point with given ID
from all provided keyframes at once
                              // --> client must find all keyframes for the
current appearance or this will cause errors!
getPoint ( $ID, $KeyFrame_ID )
getPointList ( $KeyFrame_ID )

saveKeyframe ( $data, [$recursive = false] )
addKeyframe ( $data, [$recursive = false] )
updateKeyframe ( $data, [$recursive = false] )
deleteKeyframe ( $ID )
getKeyframe ( $ID )
getKeyframeList ( $Appearance_ID )
```

General video description

```
{
   ID: "1234",
   URI: "http://yourdomain.com/content.mp4",
   W: "1280",
   H: "720",
   Duration: "254360",
   Name: "myVideo",
}
```

Representation of an object point

```
{
   ID: "0",
   X: "0.266346149444458",
   Y: "0.360165852720845",
}
```

Representation of an object time stamp

```
{
   ID: "3553",
   Hash: "6c068bf8a7c9e963995802584b4d4fec7eb7104778c0e",
   URI: "http://yourdomain.com/content.mp4",
   Type: "hd",
   Video_ID: "1234"
}
```

Representation of content related to objects

```
{
   ID: "7644",
   Name: "myName",
   Description: "myDescription",
   Tags: "",
   IconUri: "",
   WidgetName: "",
}
```

12 - CONCLUSION

In this document we have presented the architecture of the FIcontent platform. First, we have described the architecture of the three domain-specific platforms. The core of the Social Connected TV Platform (see Section 3) can be seen as a toolbox offering powerful tools to enhance connected TV services or TV related services for second-screen devices. The Smart City Platform (see Section 4) is a portfolio of functions, designed to foster the development and uptake of Smart City Applications based on Future Internet technologies. In addition, Smart City Services will be provided to showcase dedicated features of the platform. Furthermore, the Pervasive Games Platform (see Section 5) is a set of integrated, modular Enablers designed to aid building Internet-based games connected with the real world. While moving from native to in-browser execution as browser technology develops, the Pervasive Games Platform targets real-time, low latency, and high performance goals.

Second, we listed all the Generic Enabler from FI-WARE that are actively used or will be used by the FIcontent platforms. Furthermore, we explained their purpose of integration and the related Specific Enablers. This provides a brief overview about the utilized generic Future Internet technology within FIcontent.

Third, we have explained the relation between FIcontent's experimentation sites and FI-Lab regarding the deployment of FIcontent platforms. Please refer to the additional report D6.5 for more details on the deployment per scenario, application and experimentation site.

Forth, we have provided a specification of the Specific Enabler dedicated to each platform. We started with the Enablers of the Social Connected TV platform including the Audio Mining SE, Audio Fingerprinting SE, Content Optimisation SE, Second Screen Framework SE, and TV Application Layer SE. This is followed by the Open City Database SE as part of the realigned Smart City Platform. Finally, we specified the Enablers of the Pervasive Games Platformnamely the Reality Mixer SEs (Reflection Mapping SE, Camera Artifact Rendering SE), the Augmented Reality SEs (Fast Feature Tracking SE, Marker Tracking SE), the Game Synchronization SE, Spatial Matchmaking SE, and Leaderboard SE.

In addition to that, we have identified other Specific Enablers of FIcontent platforms, which are commonly used and generic rather than domain-specific. Thus, we have reclassified these Enablers (i.e. Social Network SE, Content Sharing SE, and Content Enrichment SE) as common Specific Enablers of FIcontent.Moreover, we have identified promising candidate Specific Enabler which may also become common in a later stage of the project.

The presented architecture of the FIcontent platform and the Specific Enablers are subject to change during the course of the project. Thus, the document reflects the current state and planning of the FIcontent platform. We maintain an up-to-date documentation of the architecture and the SE specification within the FIcontent Wiki [1]. Please refer to the wiki to gather recent information.

We provide an updated version of this deliverable together with upcoming milestones of FIcontent, such as the second release of the platform and Specific Enablers, and aligned to experimentation cycles, if necessary.

REFERENCES

[1] http://wiki.mediafi.org/

[2] http://wiki.mediafi.org/doku.php/ficontent.deliverables.d61

[3] http://wiki.mediafi.org/doku.php/ficontent.socialtv.roadmap.release_09_13

[4] http://wiki.mediafi.org/doku.php/ficontent.socialtv.roadmap.upcoming_releases

[5] http://wiki.mediafi.org/doku.php/ficontent.smartcity.roadmap.upcoming_releases

[6] http://wiki.mediafi.org/doku.php/ficontent.smartcity.architecture?rev=1388411732

[7] http://wiki.mediafi.org/doku.php/ficontent.smartcity.enabler.recommendationservices

[8] http://wiki.mediafi.org/doku.php/ficontent.smartcity.enabler.virtualmixedreality

[9] http://wiki.mediafi.org/doku.php/ficontent.gaming.roadmap.upcoming_releases

[10] http://unity3d.com/

[11] http://www.smartfoxserver.com/

[12] http://goo.gl/qnzjms

[13] http://catalogue.fi-ware.org/enablers/identity-management-gcp

[14] http://catalogue.fi-ware.org/enablers/identity-management-digitalself

[15] http://catalogue.fi-ware.org/enablers/identity-management-keyrock

[16] http://catalogue.fi-ware.eu/enablers/semantic-annotation

[17] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.3D-UI

[18] http://xml3d.org/

[19] http://catalogue.fi-ware.eu/enablers/object-storage-ge-fi-ware-implementation

[20] http://catalogue.fi-ware.org/enablers/iaas-data-center-resource-management-ge-ibm-implementation

[21] http://catalogue.fi-ware.org/enablers/iaas-data-center-resource-management-ge-intel-implementation

[22] http://catalogue.fi-ware.org/enablers/cdi

[23] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Middleware_Open_API_Specification

[24] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.Synchronization

[25] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.POI

[26] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.AugmentedReality

[27] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Materializing_Advanced_Middleware_and_Web_User_Interfaces_in_FI-WARE#GIS_Data_Provider

[28] http://catalogue.fi-ware.eu/enablers/location-locs

[29] http://catalogue.fi-ware.eu/enablers/s3c

[30] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.VirtualCharacters

[31] http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos

[32] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.AdvUI.AdvWebUI.CloudRendering

[33] http://catalogue.fi-ware.eu/enablers/marketplace-sap-ri

[34] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Apps.Marketplace

[35] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Apps.Store

[36] http://catalogue.fi-ware.org/enablers/access-control-tha-implementation

[37] http://catalogue.fi-ware.org/enablers/complex-event-processing-cep-ibm-proactive-technology-online

[38] http://catalogue.fi-ware.org/enablers/configuration-manager-orion-context-broker

[39] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIContent.Epic.3D-Internet%2BServices

[40] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.MiWi.AdvWebUI.InterfaceDesigner

[41] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.Epic.MiWi.AdvWebUI.2D-3D-Capture

[42] http://mediafi.org/?portfolio=use-audio-fingerprinting#tab-terms-conditions

[43] http://fi-content.iais.fraunhofer.de/fc/audiofingerprinting/#/sdk

[44] http://fi-content.iais.fraunhofer.de/fc/audiofingerprinting/#/app

[45] http://mediafi.org/?portfolio=audio-mining#tab-terms-conditions

[46] http://wiki.mediafi.org/doku.php/ficontent.socialtv.enabler.audiomining.developerguide

[47] http://wiki.mediafi.org/doku.php/ficontent.socialtv.enabler.audiomining.mediaarchiveschema

[48] http://wiki.mediafi.org/doku.php/ficontent.socialtv.enabler.audiomining.availablespeakers

[49] http://mediafi.org/?portfolio=content-optimisation#tab-terms-conditions

[50] http://wiki.mediafi.org/doku.php/ficontent.socialtv.enabler.contentoptimisation.developerguide

[51] http://mediafi.org/?portfolio=second-screen-framework#tab-terms-conditions

[52] http://www.youtube.com/watch?v=pAS5jgXlQnM

[53] http://wiki.mediafi.org/doku.php/ficontent.socialtv.enabler.secondscreenframework.developerguide

[54] https://github.com/fmtvp/tal/blob/master/LICENSE-2.0

[55] http://fmtvp.github.io/tal/index.html

[56] http://fmtvp.github.io/tal/getting-started/tutorial/installation.html

[57] https://github.com/fmtvp/tal

[58] http://fmtvp.github.io/tal/jsdoc/index.html

[59] http://wiki.mediafi.org/lib/exe/fetch.php/ficontent:smartcity:enabler:opencitydatabase:open-city-database_docu_20140324_v2.pdf

[60] http://opensource.org/licenses/MIT

[61] http://mediafi.org/?portfolio=leaderboard-se#tab-terms-conditions

[62] http://wiki.mediafi.org/doku.php/ficontent.gaming.enabler.leaderboard.developerguide

[63] http://mediafi.org/?portfolio=reality-mixer-reflection-mapping-se#tab-terms-conditions

[64] https://graphics.cg.uni-saarland.de/2013/declarative-ar-and-image-processing-on-the-web-with-xflow/

[65] http://wiki.mediafi.org/doku.php/ficontent.gaming.roadmap

[66] http://wiki.mediafi.org/doku.php/ficontent.gaming.enabler.realitymixer.reflectionmapping.developerguide

[67] http://sketchup.google.com/3dwarehouse/search?uq=1546647784845329106639920

[68] http://www.pauldebevec.com/Probes/

[69] http://en.wikipedia.org/wiki/Sphere_mapping

[70] http://mediafi.org/?portfolio=reality-mixer-camera-artefact-rendering-se#tab-terms-conditions

[71] https://github.com/fzuendeth/FIcontent.Gaming.Enabler.RealityMixer.CameraArtifactRendering/blob/master/INSTALLATION.txt

[72] http://mediafi.org/?portfolio=augmented-reality-fast-feature-tracking-se

[73] http://www.farpeek.com/index.php/apps/7-skyewars

[74] http://en.wikipedia.org/wiki/K-means_clustering

[75] http://en.wikipedia.org/wiki/Moment_(mathematics)

[76] http://mediafi.org/?portfolio=augmented-reality-marker-tracking-se#tab-terms-conditions

[77] https://developer.mozilla.org/en-US/docs/Web/API/Navigator.getUserMedia

[78] http://mediafi.org/?portfolio=spatial-matchmaking-se#tab-terms-conditions

[79] http://mediafi.org/?portfolio=game-synchronization-se#tab-terms-conditions

[80] http://wiki.mediafi.org/doku.php/ficontent.gaming.enabler.gamesynchronization.developerguide

[81] http://mediafi.org/?portfolio=social-network-enabler#tab-terms-conditions

[82] http://wiki.mediafi.org/doku.php/ficontent.common.roadmap

[83] http://wiki.mediafi.org/doku.php/ficontent.common.enabler.socialnetwork.developerguide

[84] http://wiki.apache.org/couchdb/HTTP_database_API

[85] http://wiki.apache.org/couchdb/HTTP_Document_API

[86] http://www.binpress.com/license/view/l/030cbff7f19819112b730e291abef99b

[87] http://wiki.mediafi.org/doku.php/ficontent.common.enabler.contentsharing.developerguide

[88] https://cwiki.apache.org/confluence/display/ABDERA/Getting+Started

[89] http://mediafi.org/?portfolio=content-enrichment#tab-terms-conditions

[90] http://wiki.mediafi.org/lib/exe/fetch.php/130502_fi-content2-contentenrichment.pdf

end of the document