WP5 – Content Syndication and Delivery

D5.9.1: Content Syndication and Delivery Public Report

Deliverable Lead: TIE

Contributing Partners: TIE, NTUA, UA, ASC

Delivery Date: 2015-05

Dissemination Level: Public

Final

This deliverable is the description of the current prototypes implementation of Tasks T5.2 Content Gateways, T5.3 Assets Aggregation and Composition and T5.5 Assets Marketplace, as well as the description of the second prototype of T5.4 Brand and Consumer Protection. As stated in the DOW, this deliverable identifies the descriptions of the software deliverables in WP5. This document is a living document that will be enhanced with the further delivery of the different iterations of the WP5 prototypes.

Document Status	
Deliverable Lead	TIE
Internal Reviewer 1	DW, v0.3, 04.2015
Internal Reviewer 2	TALK, v0.5, 04.2015
Type	Deliverable
Work Package	WP5 – Content Syndication and Delivery
ID	D5.9.1: Content Syndication and Delivery Public Report
Due Date	04.2015
Delivery Date	05.2015
Status	For Approval

Document History	
Versions	V0.1: TIE, Created document structure. V0.2: ALL, WP5 Partners' contributions V0.3: TIE, Reviewed version for 1st Review V0.4: ALL, WP5 Partners' contributions V0.5: TIE, Reviewed version for 2nd Review V0.6: ALL, WP5 Partners' contributions V0.7: TIE, Reviewed version for approval
Contributions	TIE: Juan Vicente Vidagany - Document creation and all sections Fran Rodriguez - Document creation and all sections NTUA: Alexandros Psyschas – Section 4 contributions UA: David Tomás – Section 5 contributions Isabel Moreno – Section 5 contributions ASC: Norman Wessel – Section 6 contributions

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.

Project Partners

TIE Nederland B.V., The Netherlands

Ascora GmbH, Germany

Talkamatic AB, Sweden

TP Vision Belgium NV, Belgium

Institute of Communication and Computer Systems, National Technical University of Athens, Greece

The University of Reading, UK

Universidad de Alicante, Spain

Deutsche Welle, Germany

Bibliographic Data Services Limited, UK

Executive Summary

This public document describes the ongoing results of the developments in WP5. It describes the different WP5 SAM Components prototypes status and results without exposing confidential information such as prototypes access information and source code. This kind of information can be shown in the related Programme Participants Prototypes (D5.2.1, D5.3.1, D5.4.2 and D5.5.1).

The SAM team decided to deliver an additional deliverable version (D5.9.0) in order to correctly describe the developments and results of the first software deliverables, especially D5.4.1 that was due in M15. Therefore, this is the 2nd version of this deliverable (D5.9.1) which contains the progress of the different prototypes until M19.

This document will be iteratively updated at the point of the delivery of the related software prototypes of the WP5 SAM Components:

- Content Gateways (T5.2 - Content Gateways)
- Linker (T5.3 - Assets Aggregation and Composition)
- Brand and Consumer Protection (T5.4 - Brand and Consumer Protection)
- Marketplace (T5.5 - Assets Marketplace)
- Syndicator - Data API Services (T5.5 - Assets Marketplace)

At the moment, the deliverable contains the following software prototype information:

Task	Component	Software Prototype Deliverable	Due	Section
T5.2	Content Gateways	D5.2.1	M19	3
T5.3	Linker	D5.3.1	M19	4
T5.4	Brand and Consumer Protection	D5.4.2	M19	5
T5.5	Marketplace + Data API Services	D5.5.1	M19	6

For each of the SAM Component prototypes, the following information is presented:

- **Scope and Relationship:** Describes the scope of the component implementation, its purpose and the main relationships with other modules being implemented in SAM
- **Requirements and Preparations:** Introduces the information needed to deal with the prototype, in terms of technical and non-technical requirements, software to be installed, etc
- **Installation:** Describes the steps needed to install the software, and how to build it from source code
- **Execution and Usage**: Presents the different screens and actions implemented in the prototype itself, how to access it and how to test the different implemented options
- **Limitations and Further developments:** Depicts the current prototype limitations and the expected improvements in the next iteration
- **Summary:** Describes the conclusions of the implementation of this prototype

Table of Contents

1 Introduction

SAM – Dynamic Social and Media Content Syndication for 2nd Screen – is a project funded by the Seventh Framework Programme of the European Commission under Grant Agreement No. 611312. It provides a content delivery platform for syndicated data to be consumed in a contextualised social way through 2nd Screen devices.

1.1 SAM Project Overview

The current generation of Internet-connected devices has changed the way users interact with media. Previously, users were restricted to being passive and unidirectional consumers; now, they are proactive and interactive media users. They can comment on and rate a television show or film and search for related information regarding cast and crew, facts and trivia or even filming locations. They do this with both friends and wider social communities through the so-called '2nd Screen'.

Another related phenomenon is 'Content Syndication', which is a field of marketing where digital content is created once and delivered to consumers through various different marketing channels (devices, markets and stakeholders) simultaneously, enabling efficient content control, delivery and feedback.

However, the 2nd Screen phenomenon has grown in a disorderly manner. Tools supplied by the media provider companies (e.g. as mobile or tablet apps) limit the potential outreach and, as a result, users are not enjoying relevant contextual syndicated information. European enterprises wishing to provide services have limited methods of receiving feedback, restricting the business intelligence that can be extracted and applied in order to profit from and enrich this growing market.

SAM is reshaping the current disorganised 2nd Screen ecosystem by developing an advanced social media delivery platform based on 2nd Screen and Content Syndication within a social media context. This is achieved by providing open and standardised means of characterising, discovering and syndicating media assets interactively. Users will be able to consume and prosume digital assets from different syndicated sources and synchronised devices (e.g. connected televisions), creating more fulfilling experiences around the original media assets.

The SAM vision that is now becoming reality, sees the former, out-dated system of users searching for the information they desire replaced with a new approach where information reaches users on their 2nd Screen using content syndication. This is enriched through the creation of dynamic social communities related to the user and digital asset context (e.g. profiles, preferences and devices connected). These are continuously evolving social spaces where people share interests, socialise and build virtual communities. SAM will enable syndication of comments, ratings, facts, recommendations and new information that will enrich and energise the virtual community as well as enhance personalised knowledge and satisfaction.

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to accompany the software prototypes of WP5 tasks T5.2 Content Gateways, T5.3 Assets Aggregation and Composition, T5.4 Brand and Consumer Protection and T5.5 Assets Marketplace. Each task will contribute different Components to the SAM Architecture that are developed iteratively in 3 phases as per

milestones 3/4/5 at M19/25/31 and will produce a software deliverable and an update of this public documentation (D5.9).

As the main focus of the tasks is the development of the software itself, this accompanying document focuses on providing a short summary of the main functionalities and on serving as a user guide for the current status of the development.

1.3 Document Status and Target Audience

This document is the second deliverable pertaining to the D5.9.x series (since an extra deliverable D5.9.0 was delivered in M15). They are listed in the DOW as public, since it presents public information related to the prototypes of the software components in WP5 tasks.

1.4 Abbreviations and Glossary

A definition of common terms and roles related to the realisation of SAM, as well as a list of abbreviations, is available at http://wiki.socialisingaroundmedia.com/index.php/Glossary.

1.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1 (Introduction):** Provides an overview of the entire document and the related pilot implementation, describing the objectives, constraints and status
- **Section 2 (WP5 Introduction):** Provides an overview of WP5 goals and SAM Components
- **Section 3 (Content Gateways):** Describes the latest software deliverable developed in T5.2
- **Section 4 (Asset Aggregation and Composition):** Describes the latest software deliverable developed in T5.3
- **Section 5 (Brand and Consumer Protection):** Describes the latest software deliverable developed in T5.4
- **Section 6 (Assets Marketplace):** Describes the latest software deliverable developed in T5.5
- **Section 7 (Document Summary):** Briefly summarises the work presented in the deliverable as well as the overall WP5 status

In Sections 3 to 6, for each component in the SAM Architecture, the following subsections are provided:

- **Scope and Relationship:** Describes the scope of the component implementation, its purpose and the main relationships with other modules implemented in SAM in the first year.
- **Requirements and Preparations:** Introduces the information needed to deal with the prototype in terms of technical and non-technical requirements, software to be installed, etc.
- **Installation:** Describes the steps needed to install the software, and how to build it from source code.
- **Execution and Usage**: Presents the different screens and actions implemented at the prototype itself, how to access it and how to test the different implemented options.

- **Limitations and Further Developments:** Depicts the current prototype limitations and the expected improvements.
- **Summary:** Describes the conclusions of the implementation of the prototype.

1.6 External Annexes and Supporting Documents

- D5.2.1_Content_Gateways_(1st_Prototype)_For_Approval (PP)
- D5.3.1_Asset_Aggregation_And_Composition_(1st_Prototype)_For_Approval (PP)
- D5.4.2_Brand_and_Consumer_Protection_(2nd_Prototype)_For_Approval (PP)
- D5.5.1_Assets_Marketplace_(1st_Prototype)_For_Approval (PP)

2 WP5 Introduction

WP5 focuses on the structure, syndication and distribution of Assets in SAM. It includes the mechanism to describe, import, link, filter and deliver the information in the SAM Platform. The integration of external content as well as its correct distribution is a crucial element for SAM, importing already existing external content will speed up the growth of asset volume and its value for business users.

Specific objectives of this WP include:

- To define an open Asset format and Open Specification API so that it can be shared with other initiatives dealing with 2nd Screen technologies or entertainment data (T5.1, T5.5)
- To provide gateways and semantic integration techniques to import contents from 3rd party systems (T5.2)
- To implement editor tools to compose, aggregate and annotate content and services assets (T5.3)
- To implement filtering mechanisms for Brand and Consumer Protection (T5.4)
- To establish a Marketplace of contents and services (T5.5)

The result of WP5 will be a set of related content-syndication orientated components (see Figure 1) that will enable source media information to be shared and controlled whilst ensuring integrity. Each component will be developed iteratively in 3 phases as per milestones 3/4/5 at M19/25/31 and will produce a software deliverable as well as an update of the public documentation (D5.9.x – this document series).

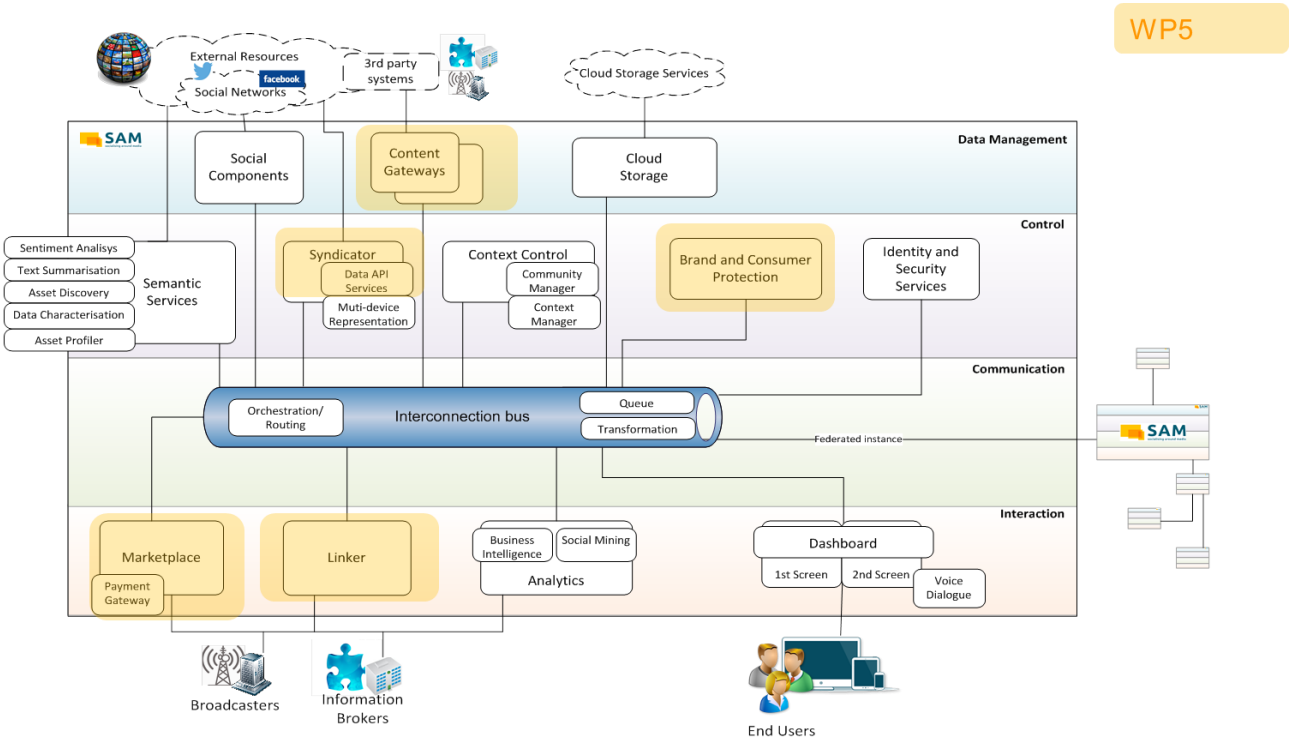

Figure 1: WP5 Components Contribution to SAM Architecture

The work in WP5 (as well as in the other development WPs) is being managed by using the agile methodology SCRUM. For that purpose, a specific WP5 SCRUM board has been created in the SAM Jira, being the Scrum Master a representative of the WP Lead (TIE).

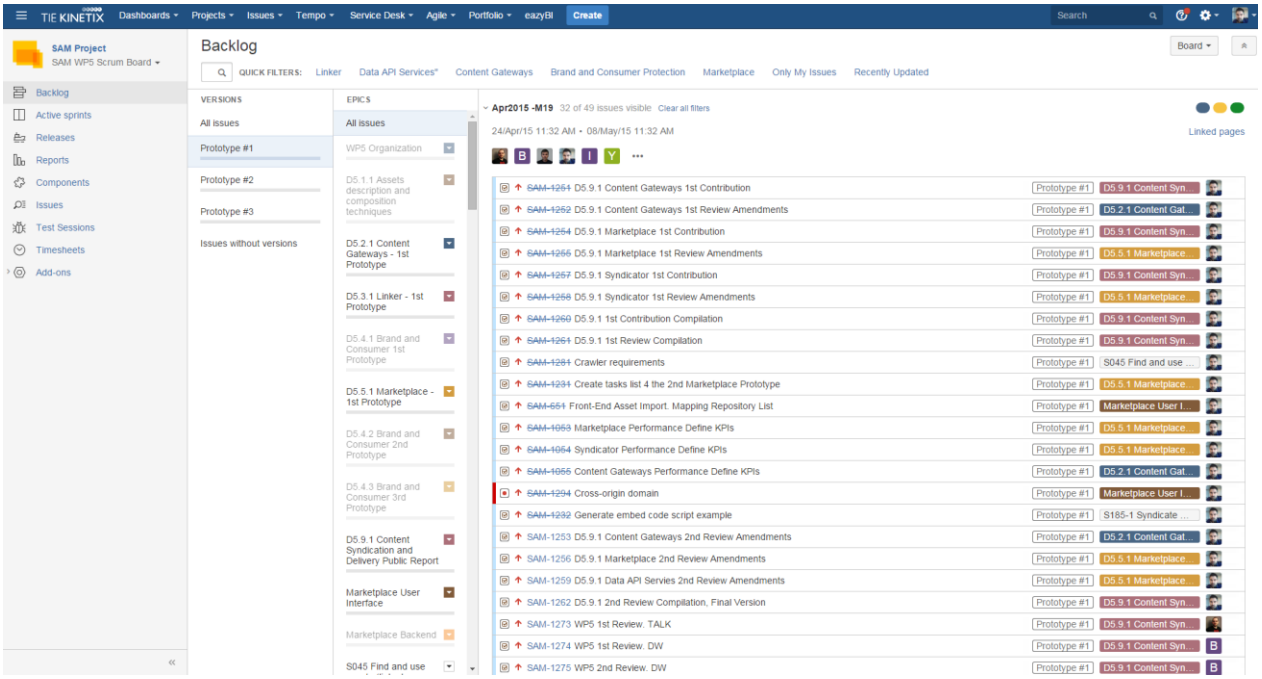

Figure 2: WP5 Scrum Board

Sprints are planned monthly. Every story or task is linked to one or more specific requirements as expressed in D2.3 (User Stories and Requirements).

A Planning meeting is scheduled at the beginning of each sprint in order to plan the next monthly sprint and discuss the priorities or reschedule the work not finished in the previous one. A Retrospective meeting is also scheduled at the end of each sprint in order to discuss the work done during the sprint and find ways of improving (if necessary) the way of working.

3 Content Gateways

This section describes the software deliverable D5.2.1, which is the first prototype release of the SAM Content Gateways component.

3.1 Scope and Relationship

The Content Gateway is the component in charge of data gathering from external data sources and 3rd party systems. The objective is to implement strategies, tools and techniques to allow the easy integration of heterogeneous content sources into the SAM Platform.

To carry out the communication and data gathering from external systems, the component is broken down into several subcomponents, which can be seen in the Figure 3.

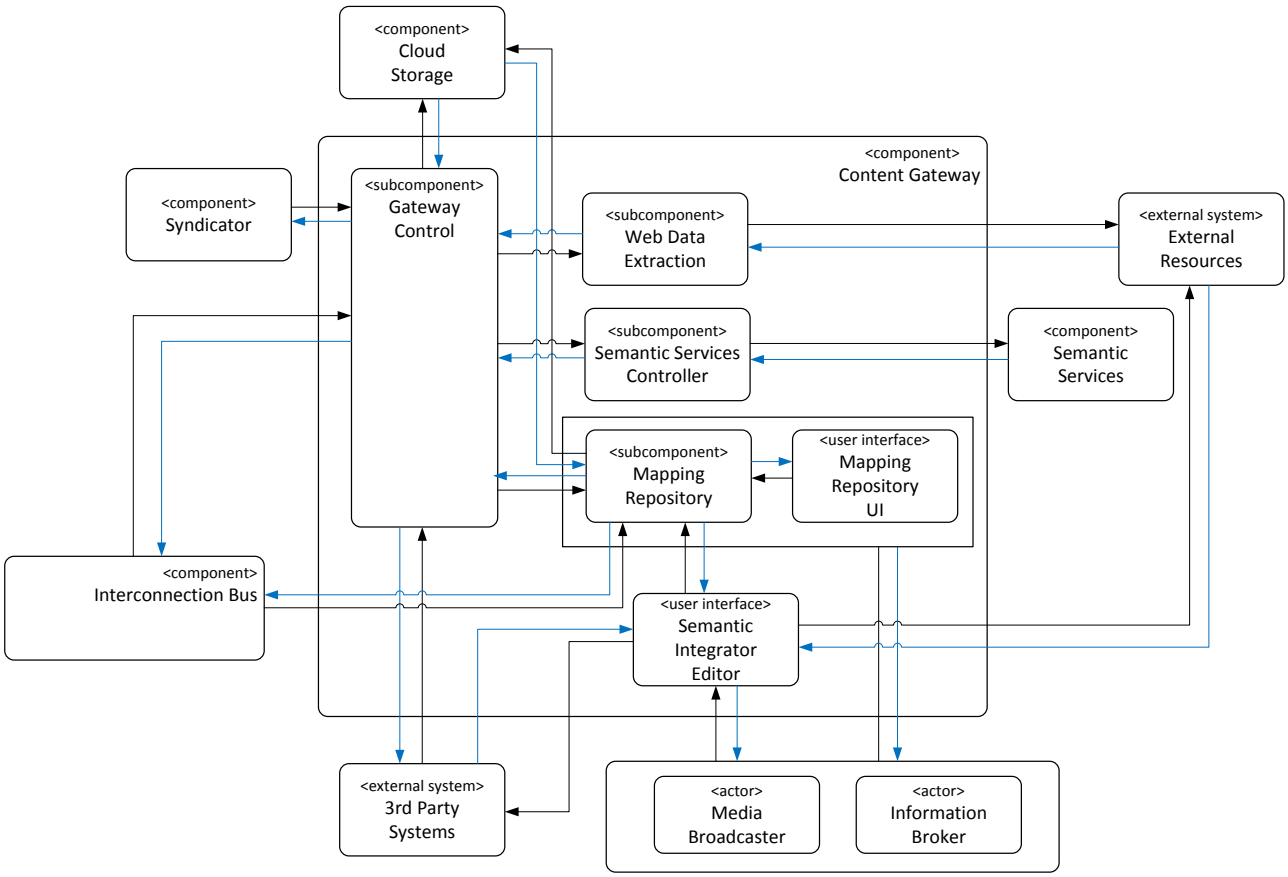

Figure 3: Content Gateway

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.5 (Architecture), D3.2.2 Section 4.6 (Functional Specification) or D3.3.1 Section 3.6 (Technical Specification).

Based on these documents, the Semantic Integrator Editor has been implemented using a semantic mapping approach by adapting and extending the TIE Semantic Integrator product (TIE TSI). And, in order to develop the Mapping Repository, a web application has been implemented where the TSI mapping files can be stored along with their description information. The Mapping Repository uses its own storage and search mechanisms which will be able to share and synchronise the information with SAM native storage systems.

The different technologies involved in this implementation are:

- Scala and Play framework – Play has been chosen as the Model-View-Controller (MVC) framework and Scala as the implementation language
- MongoDB and GridFS: The mapping description is saved in a MongoDB and the actual attachments are saved in GridFS
- ElasticSearch: The Mapping uses Elastic Search, a search engine based on Apache Solr[1], that allows fast data finding and full text search (even in the file attachments), additionally proving index based search and a full query language
- Akka Actors: Akka actors are used for concurrent processes and each service has an actor attached to it to avoid concurrency issues as actors run asynchronously, improving in that way the performance and robustness of the solution.

The Mapping Repository has been implemented using service interfaces defined separately for each module or functionality (e.g. search, store, etc.). It also stores metadata information in its own MongoDB storage (as in this prototype it is not yet integrated with the Cloud Storage). Each map object in the Mapping Repository contains the metadata information as attributes and mapping files are stored as attachments in the MongoDB repository as a GridFS attachment.

As part of the Content Gateways development, Semantic Services provides a RESTful interface for suggesting alignments between the SAM ontology and the structure of the contents that will be imported into the platform by the Content Gateways. The documentation demo page for this service can be shown in the deliverable D4.3.1 and further information can be found in D4.9.1.

Finally, a 1st version of the interface to configure the importations has been integrated into the Marketplace. End Users will use this interface based on the previous mapping carried out. This task is part of the communication with the Interconnection Bus component which will be in charge of executing these importations. See section 6.1.4.1.7 for further details.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table (Figure 44).

Subcomponent	Task
Semantic Integrator Editor	• SAM TSI new installation • Develop new mapping recommendation algorithms to improve the suggestions • Add new functionality in order to add parameters in the LPO generation • Start to work in the transformation from BDS structure to Syndicator structure
Mapping Repository	• 1st version of the Mapping Repository web interface • Implementation of the necessary web services and the engine to use Elastic Search and the own MongoDB instance
Gateway Control	1st Asset Import version integrated in the Marketplace

Figure 4: Tasks Carried Out for the First Prototype of T5.3

3.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users and developers, as well as software prerequisites for the installation of the Linker component.

[1] https://lucene.apache.org/solr/

3.2.1 Semantic Integrator Editor

The following prerequisites apply for both users and developers before attempting to install and use the TSI tool:

- Windows XP/7 (32/64-bit)
- At least 500 MB of available hard drive space and a minimum of 512 MB RAM
- Admin rights on the machine to create/edit files
- Install Java Runtime Environment (JRE) [2]
- Install Apache Tomcat[3]
- Install only for developers Eclipse Luna IDE [4]

3.2.2 Mapping Repository

3.2.2.1 For Users

The user should use any of the most popular browsers (Chrome, Firefox, Internet Explorer, Safari). The development and primary testing has been done using Chrome.

3.2.2.2 For Developers

- At least 500 Mb of available hard drive space
- Install MongoDB[5]
- Install Play Framework 2.2.1 [6]

3.3 Installation (Deployment)

This section will describe the process of installing and configuring the Semantic Integrator Editor and the Mapping Repository intended for the user. It will go through the various configurations and settings that have to be performed by the user before starting to use both tools.

3.3.1 Semantic Integrator Editor

The URL needed for downloading the Semantic Integrator Editor can be found in the deliverable D5.2.1. It can be installed using the .exe installer file, after unzipping the downloaded file. In the figures below, the window dialogs are shown for configuring the different components. Once they are executed, the Wizard installer will start.

[2] http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html
[3] http://tomcat.apache.org/
[4] https://www.eclipse.org/luna/
[5] https://www.mongodb.org/
[6] https://www.playframework.com/download#older-versions

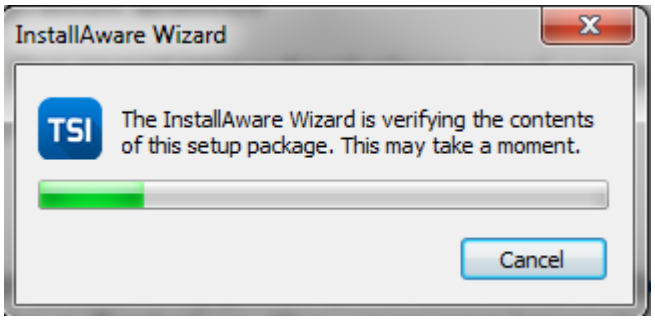

Figure 5: Install Aware Wizard SAM TSI

1. Click on the “Next” button

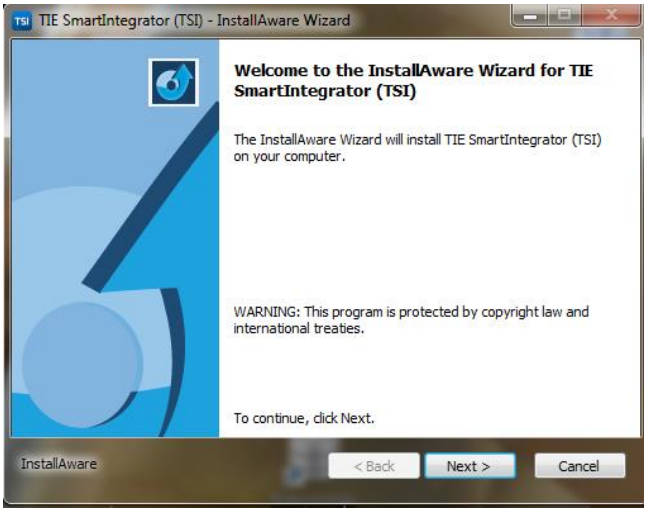

Figure 6: Welcome to the Installation SAM TSI

2. Accept the License agreement and click on the “Next” button

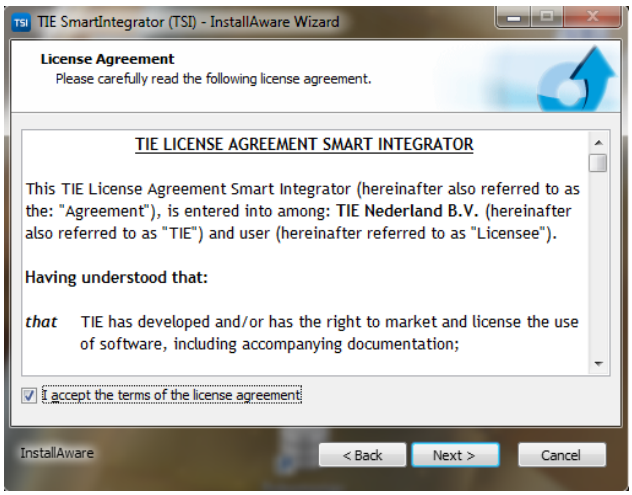

Figure 7: License Agreement SAM TSI Installation

3. Introduce User Name and Organisation

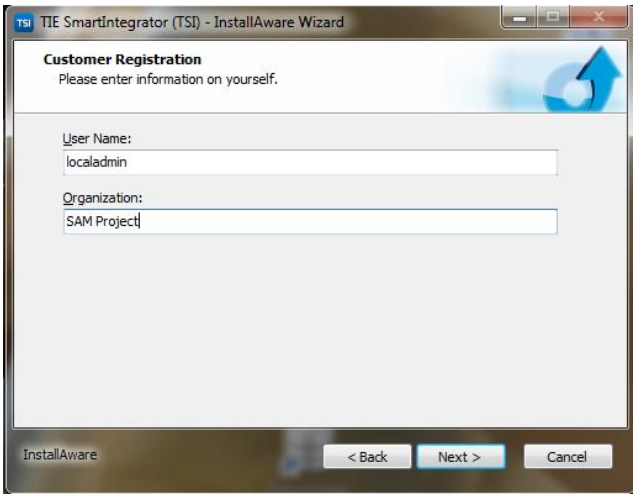

Figure 8: Customer Registration SAM TSI Installation

4. Select Complete Installation

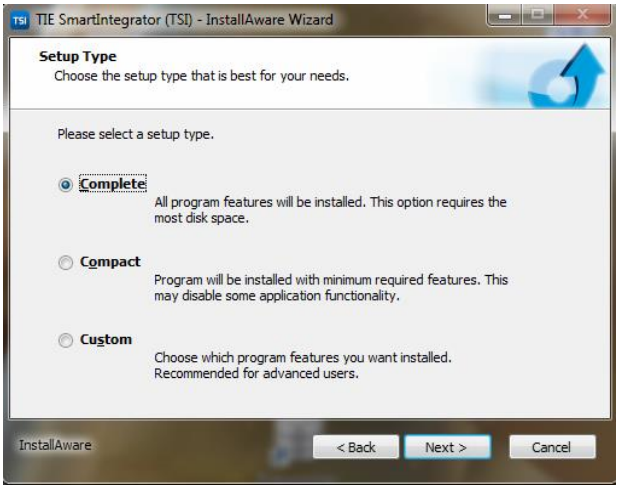

Figure 9: Setup Type SAM TSI Installation

5. Select the folder where setup will be installing files

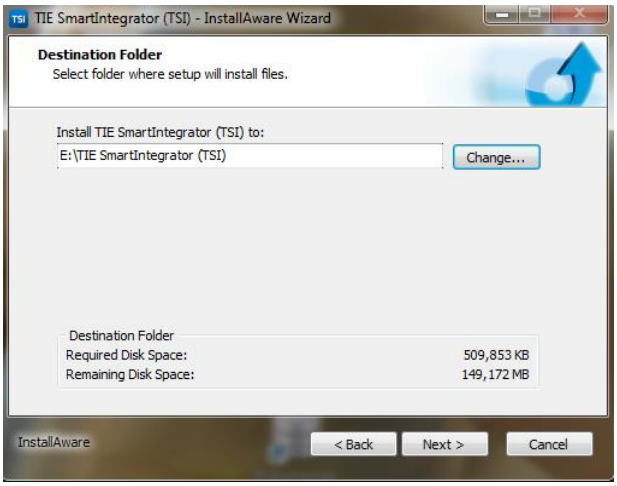

Figure 10: Destination Folder SAM TSI Installation

6. Select the Program Folder and install the application for anyone who uses this computer

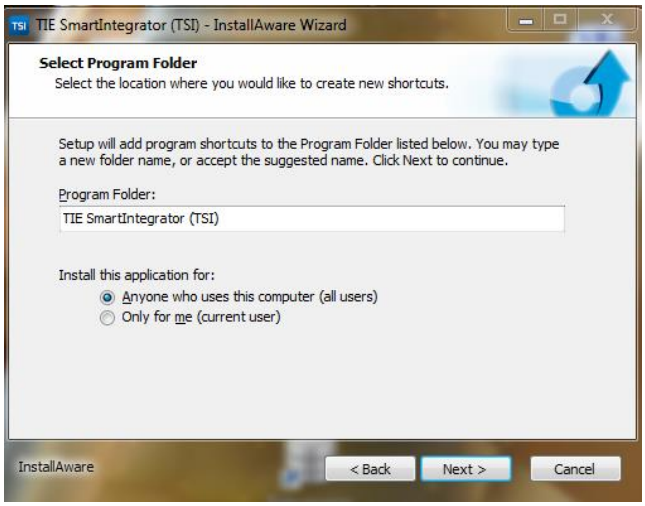

Figure 11: Program Folder SAM TSI Installation

7. To complete the installation click on the “Next” button

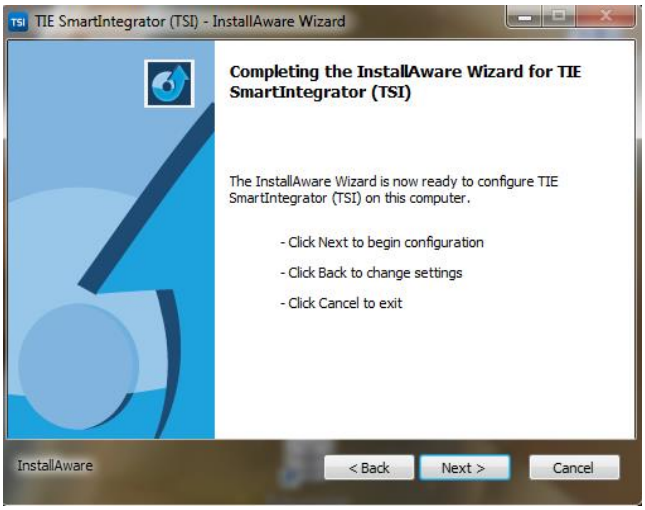

Figure 12: Completing Installation SAM TSI Installation

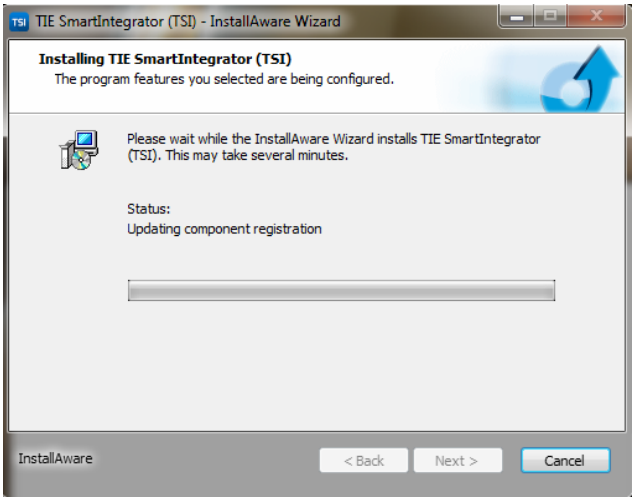

Figure 13: Completing Installation SAM TSI Installation

8. Finish the installation

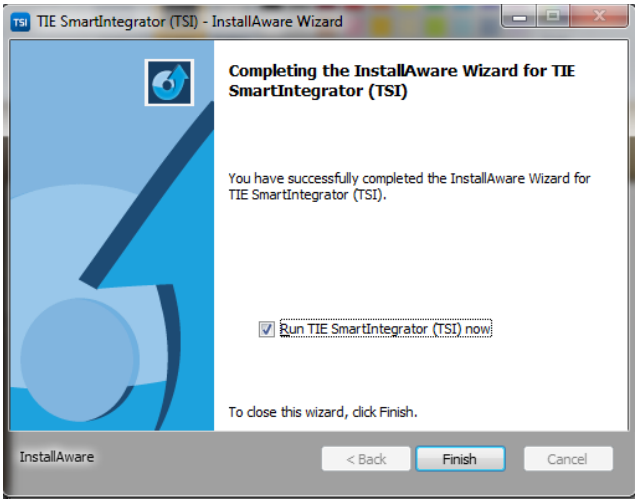

Figure 14: Finish Installation SAM TSI Installation

3.3.2 Mapping Repository

Currently the installation is carried out by the Jenkins Continuous Integration Server[7] provided by the SAM consortium. For the Mapping Repository, a Jenkins integration project has been created and configured to build and deploy the mapping repository in a specific Play Framework folder.

In order to start the execution of the mapping repository the following scripts have been created:

- Start.sh
- Stop.sh
- Deploy.sh

3.4 Execution and Usage

This section introduces the required steps to execute and use the Semantic Integration Editor (Section 3.4.1) and the Mapping Repository (Section 3.4.2).

3.4.1 Semantic Integrator Editor

Over the next sections, the most important steps and options in the usage of Semantic Integrator Editor are explained.

1. Double-click the TSI icon installed in the desktop

Figure 15: Shortcut SAM TSI

2. Configure the Preferences

[7] http://jenkins-ci.org/

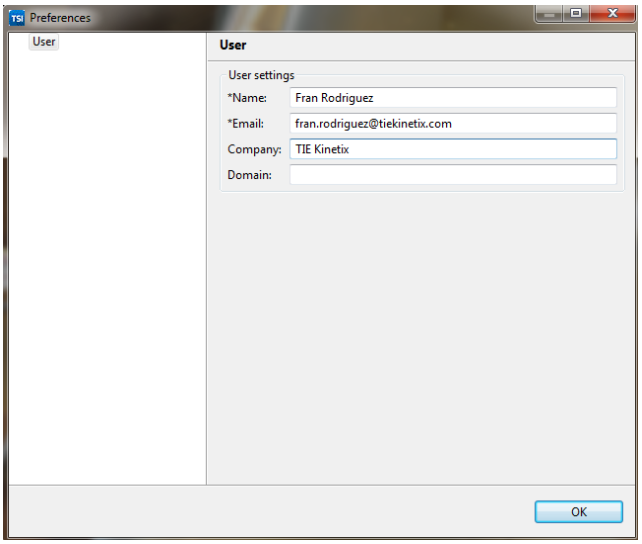

Figure 16: User Preferences SAM TSI

3. Click on “TSI Workbench”

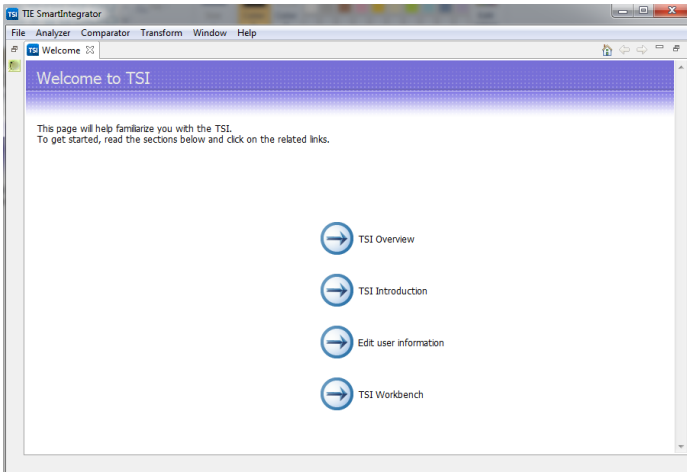

Figure 17: Welcome SAM TSI

3.4.1.1 Basic Steps

With TSI you can map and transform data from one format to another. To accomplish that, the basic steps in TSI are:

- Import the first data file and make sure that TSI can read the data
- Import the second data file and make sure that TSI can read the data
- Map and transform the data, using a side-by-side view (the Comparator)
- Publish your mapping file (called LPO - Linked Plan Object). The resulting mapping file is actually a file that you can use in scripts to map and transform files

3.4.1.2 Start a Project

For every Content Provider project it will be probably necessary to create multiple mappings. Therefore, start by creating a project that will serve as an umbrella for mappings that belong to one project.

The steps to create a project are:

1. Inside the TSI Dashboard, click “Create” from the “New Project”, as shown in the diagram below. It is also possible to do this from the menu “File” → New → TSI Project

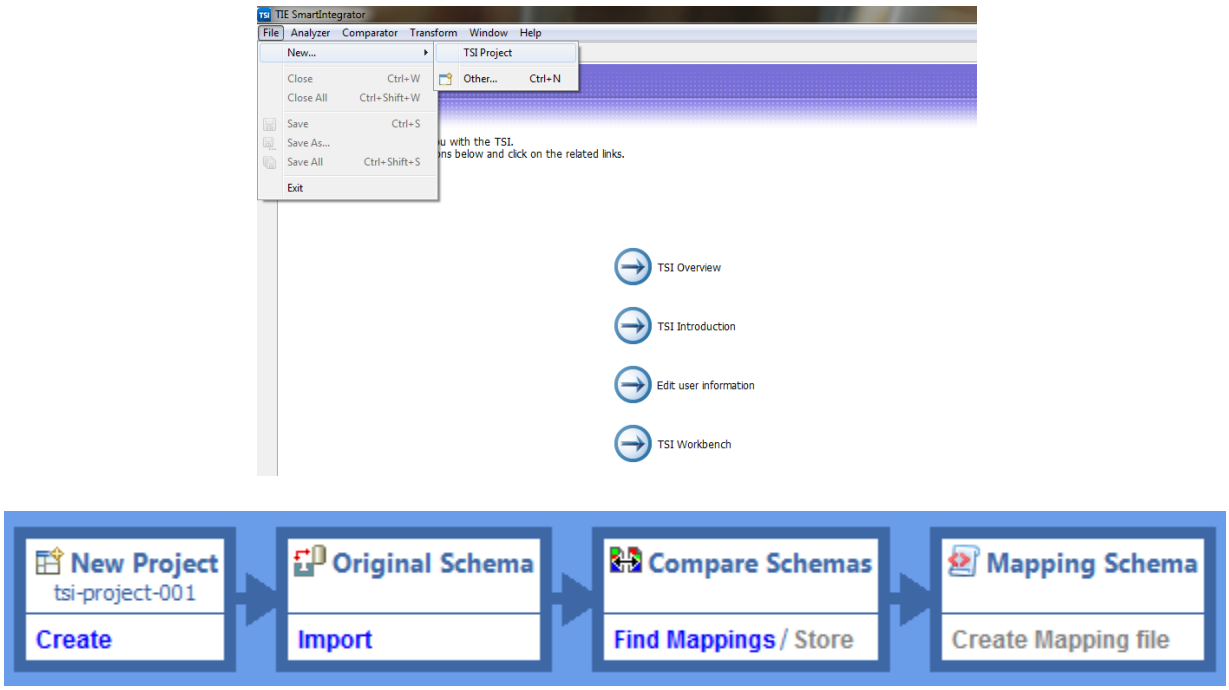

Figure 18: Start Project SAM TSI

Enter a descriptive name for your project

a. For this first project make sure “Import original schema” is selected

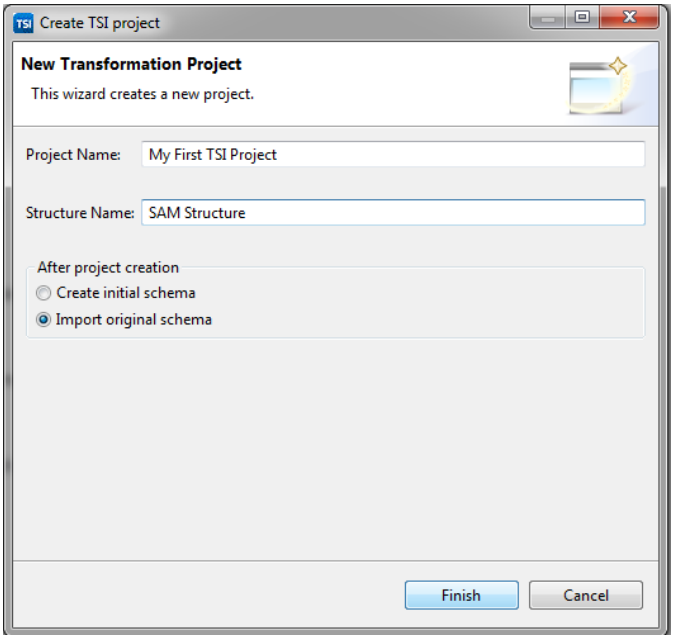

Figure 19: Create SAM TSI

3. Click finish

You will have a project folder created, with a subfolder for every mapping

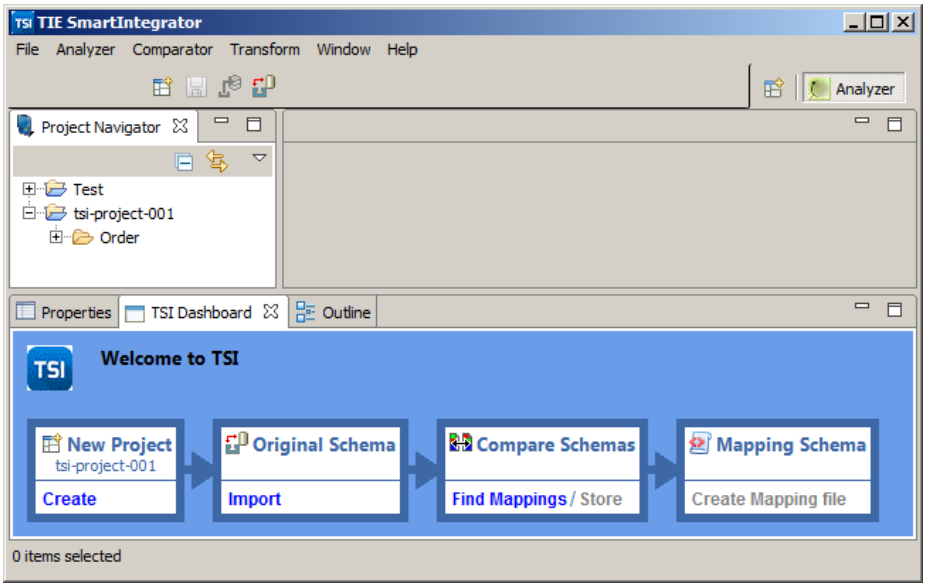

Figure 20: SAM TSI Dashboard

3.4.1.3 Importing Files

TSI can map from and to numerous types of file formats, and every format has multiple flavours. Therefore, TSI has different import settings for every format. This section explains the import settings which apply for all formats.

The general procedure for importing files is:

1. In the TSI primary flow diagram, click on ‘Import’ (i.e. the second step)

Figure 21: SAM TSI Flow

2. Select the file format

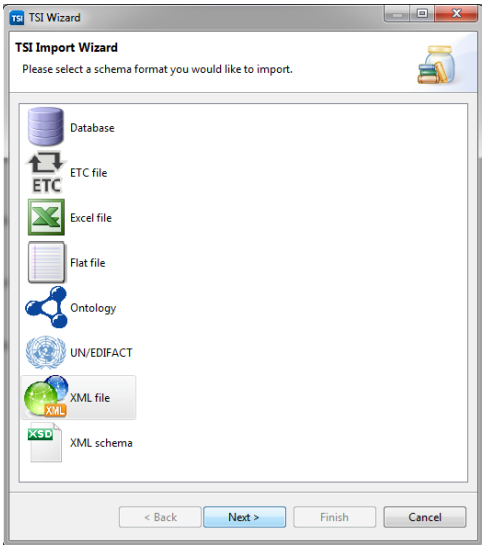

Figure 22: Select File Format SAM TSI

3. Select the file to be imported, using the ‘Browse’ button

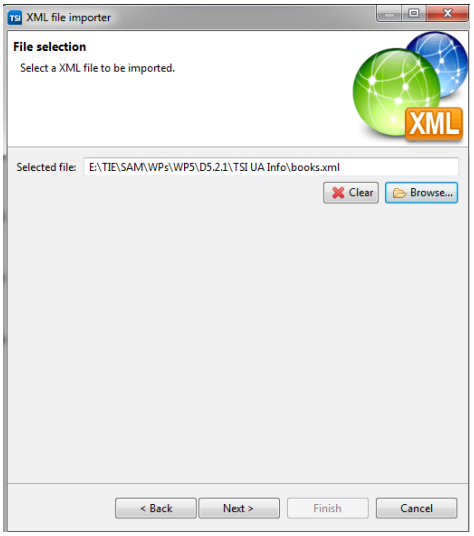

Figure 23: File Selection SAM TSI

4. Configure file specific options
5. Indicate to which mapping folder you would like to save your imported file. Select a subfolder of the project for which you will use the file, and click Finish

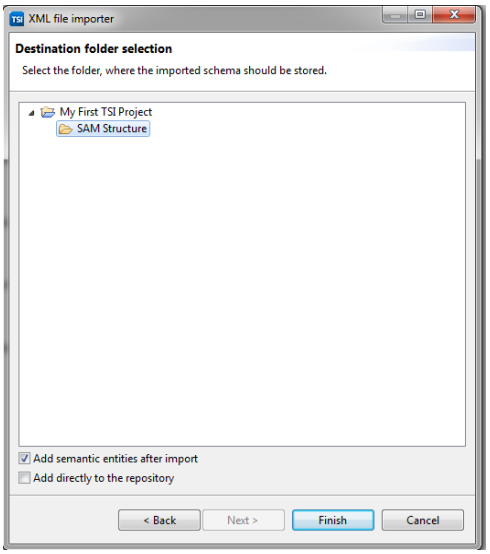

Figure 24: Destination Folder Selection SAM TSI

TSI will then import the file, analyse its contents, and create a visual representation of its contents. TSI will also provide a human-readable representation of the original element, displayed as a second row of elements.

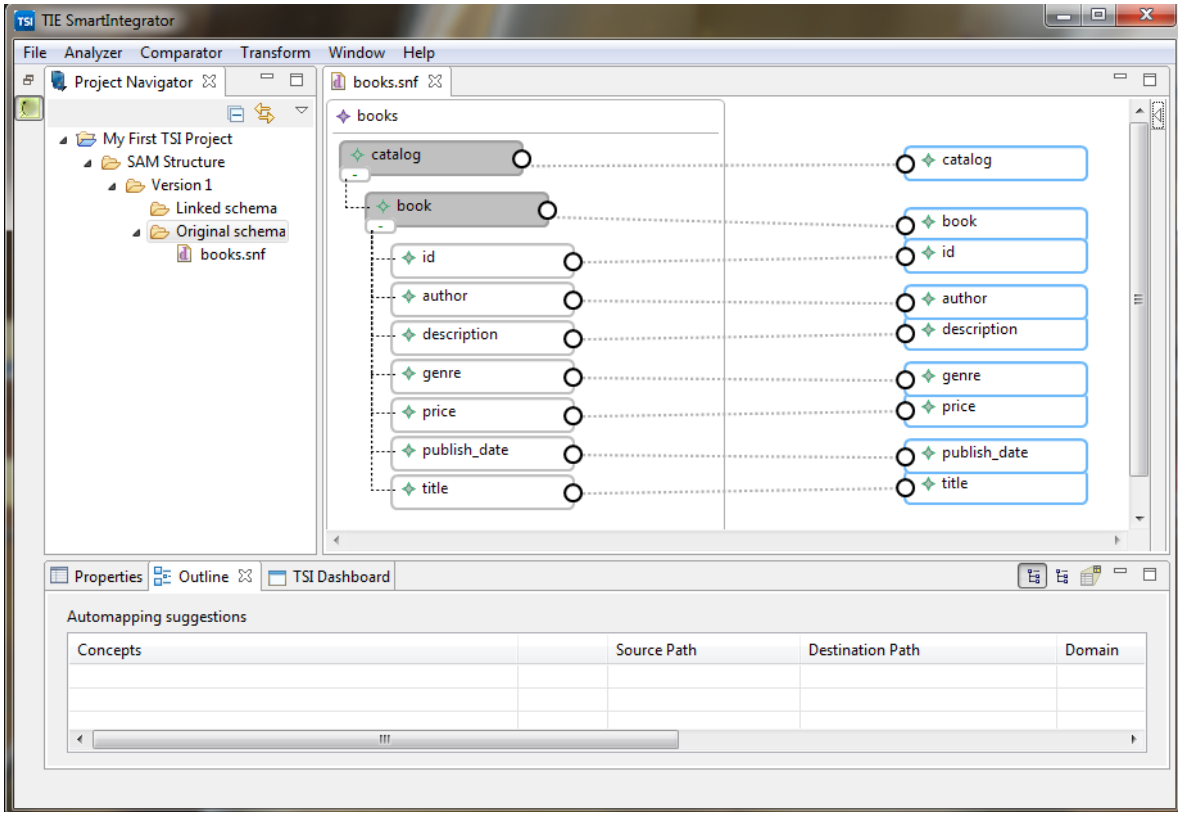

Figure 25: Project Navigator SAM TSI

In the section about importing files you will have probably noticed that all files show a secondary column with elements in blue balloons.

- The left column contains the original elements that carry their syntactical names. For instance '_0004'
- The right column contains the human-readable version of the original names, which are their semantic names, for instance 'sender identification number'

The semantic names are meant to facilitate the mapping process.

3.4.1.3.1 Saving Your Work

There are two ways for saving your work:
1. Save the current state of your unfinished work. To the current state of your work, simple use the CTRL+S shortcut, or the file menu.

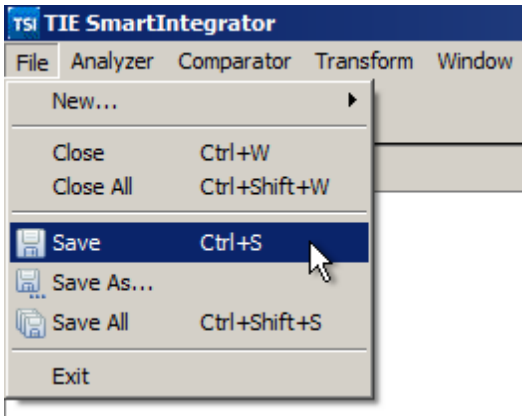

Figure 26: Save SAM TSI

2. Saving (the final version of) your imported files to the repository, to be used for the mapping. You are ready to save your file to the repository when you have successfully imported your file, you have a correct graphical representation of the file, and you have finished editing the file if necessary.

To save your file to the repository:
1. Click somewhere in the diagram of the file you want to store
2. From within the TSI primary flow diagram, click on 'Store', as shown in the figure below

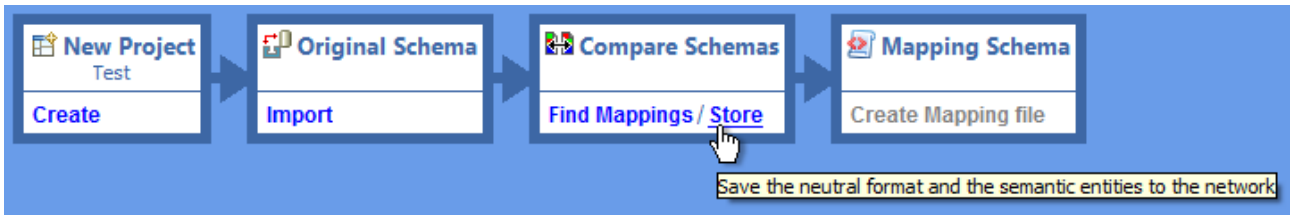

Figure 27: Store SAM TSI

Alternatively, click the 'Save to Repository' button in the navigation bar above the canvas:

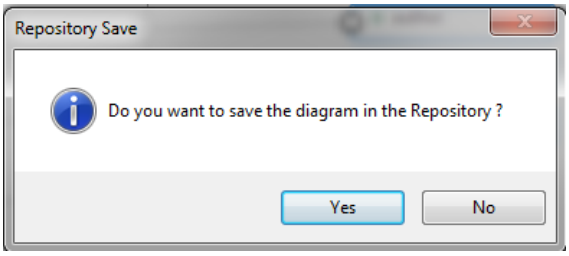

Figure 28: Save to Repository SAM TSI

3.4.1.4 Mapping: Comparing Imported Files and Pairing Corresponding Elements

After storing the files in the repository, the files are ready for the mapping. With TSI you can create a mapping in a graphical way, by showing two files on screen that need to be mapped. You are then able to pair corresponding elements of the two files with each other, by drawing lines between these elements:

3.4.1.4.1 Import a New File

This process is explained in section 3.4.1.3

3.4.1.4.2 Select Two Files for the Mapping

To select the files you want to use in the mapping, start the "Comparator Wizard" for comparing two schemas:
1. Click "Comparator" from the top menu and then click "Compare TSI Schema"

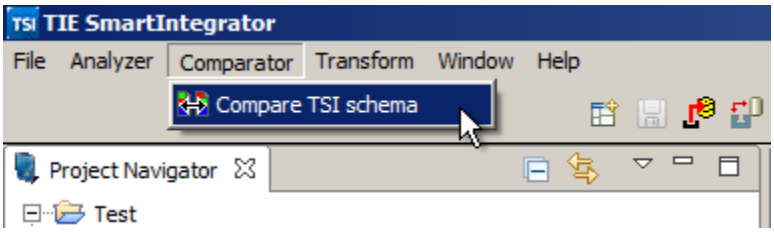

Figure 29: Compare Schema SAM TSI

2. You will see a wizard like the one shown in the figure below

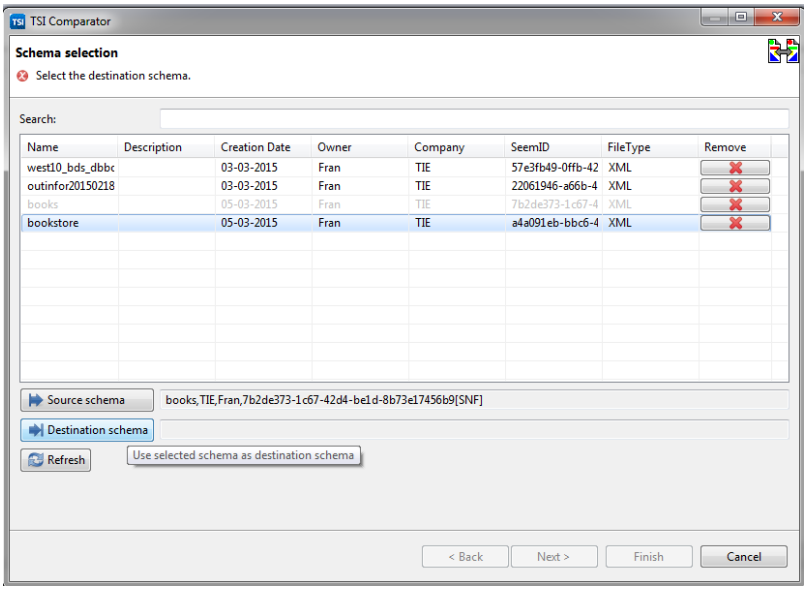

Figure 30: Select Schemas SAM TSI

3. All the stored schemas in the Repository are listed here (if you don't see the recently stored schema, click the "Refresh" button")
4. Select the "Source" schema first (you can double click on the selected row)
5. Do the same for the Destination schema
6. After selecting the source and destination schema, click "Next"

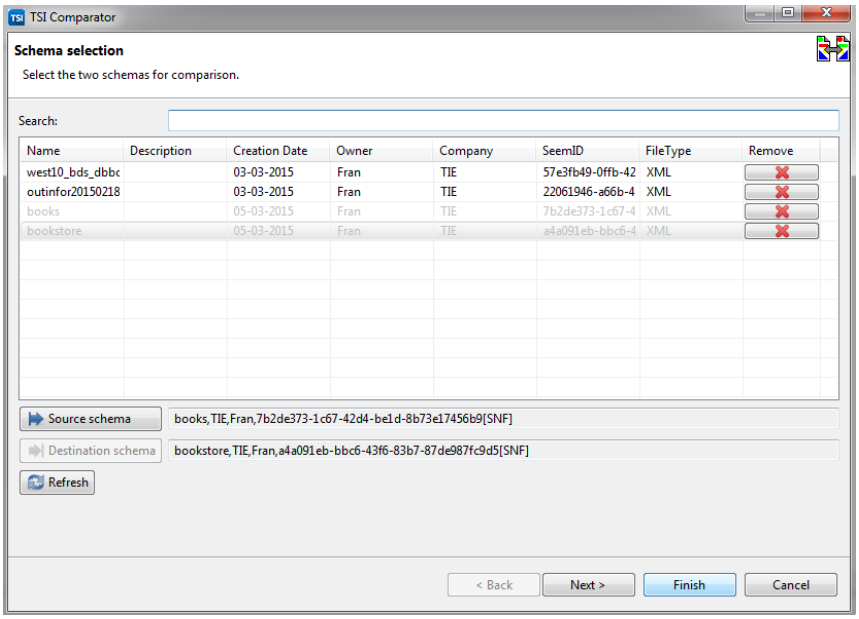

Figure 31: Finish Schemas Comparison SAM TSI

7. Select the Project where you want to store the comparator diagram, for example as shown in the figure below. For mappings, select the ‘Linked schema’ folder

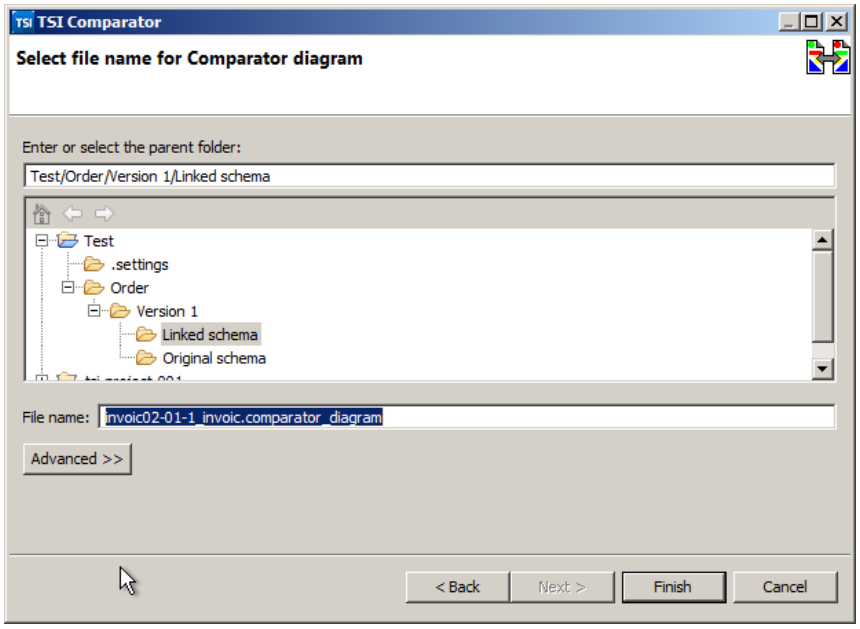

Figure 32: Linked Schema SAM TSI

8. Click “Finish” and you will see the comparator diagram on the editor

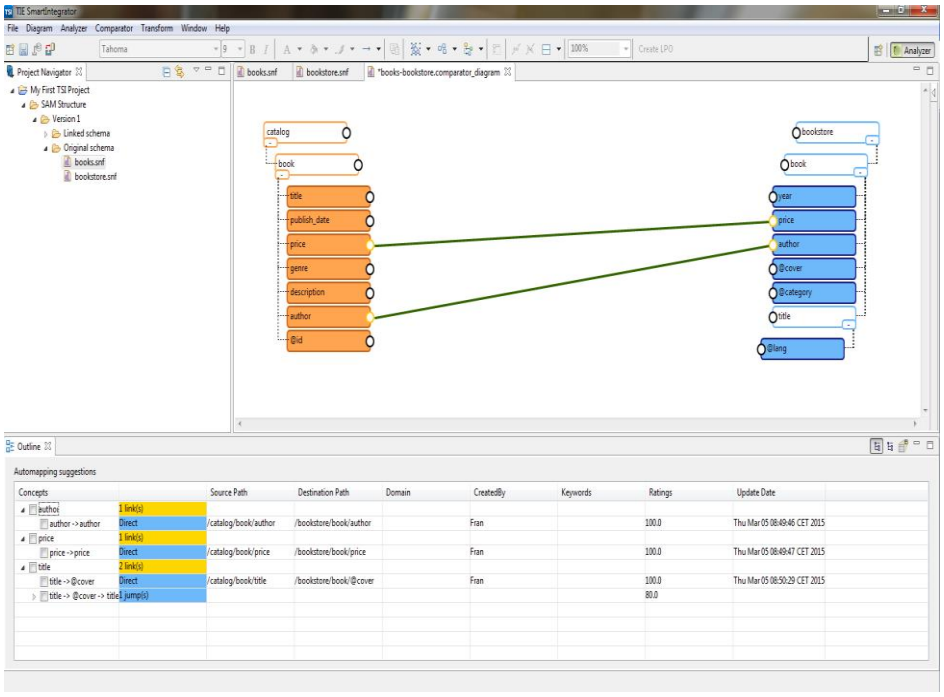

Figure 33: Make relations SAM TSI

3.4.1.4.3 Map the Corresponding Elements

Now you have two files in the comparator view, it is time to link the corresponding elements to each other.

1. Locate a set of elements that should be mapped to each other

2. Click on the Link button in the Objects panel (right-hand sidebar)
3. Drag a line from the source element to the destination element
4. Repeat steps 1-3 for a number of elements, then create a test mapping file (see further on in this user guide), and try your partial mapping on a test file
5. If that works, repeat steps 1-4 until you linked all the details that your situation requires

3.4.1.5 Creating a Mapping File

By using the comparator diagram you can create a java based mapping file, called Linked Plan Object (LPO). In order to create the mapping file, please follow the steps below:

1. Open the Comparator Diagram (if not already open) with a mapping
2. Save the diagram after doing any changes
3. Click on the icon “Create LPO file” as shown in the figure below

A file browser screen will open. Select the data file that you used for the target data:

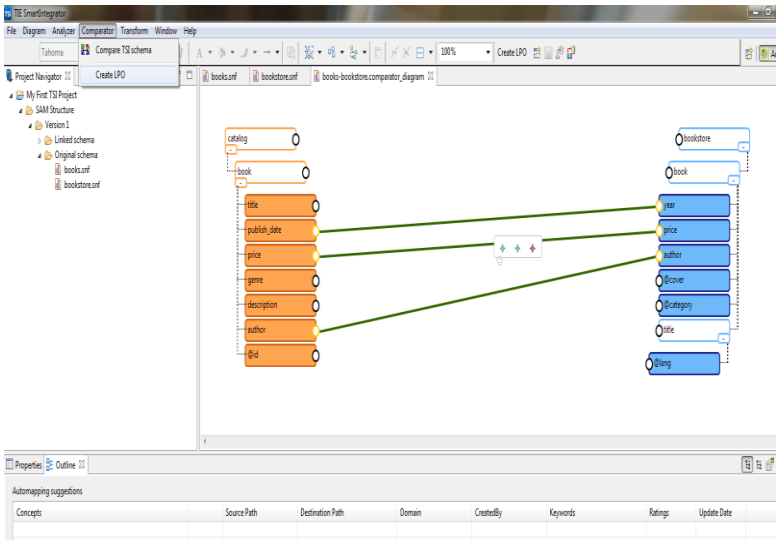

Figure 34: Creating Mapping File SAM TSI

3.4.1.6 Transform the Mapping File

To transform the mapping file, go through the following steps:
1. In the horizontal top menu, go to Transform > Select the files
2. You will be prompted with a dialogue to select/configure certain file, as shown in the figure below
3. Select the relevant files by clicking on “Change”. Once all the files are chosen, click “Run” to execute the LPO
4. A Progress bar will demonstrate the transformation process, and once the transformation is done, you will be notified with a message

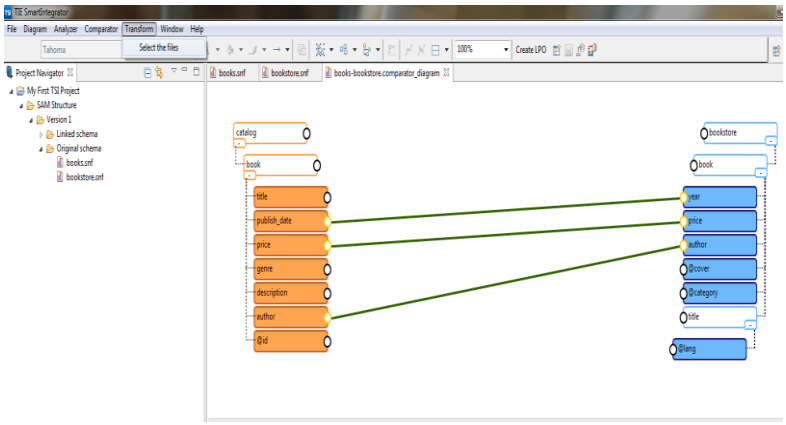

Figure 35: Transform Mapping File SAM TSI

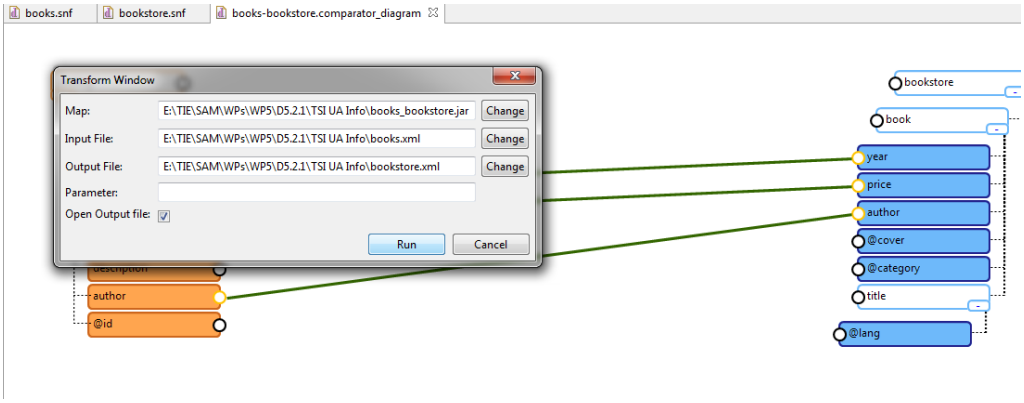

Figure 36: Run Transformation Mapping File SAM TSI

3.4.2 Mapping Repository

Over the next sections, the most important steps and options in the usage of the Mapping Repository are explained.

3.4.2.1 Dashboard

The Dashboard shows the available mappings already created. It is also possible to filter them by introducing a free text.

Figure 37: Mapping Repository Dashboard

3.4.2.2 Create Mapping

Click on the icon “New” in order to create a new mapping. The following information should be introduced:

1. Name. It is a text field to introduce a descriptive name of the mapping
2. Engine. It is a combo box field, where the user should select the LPO option
3. Version. It is a text field to introduce the number of the version of the repository, for instance v0.1
4. Url. It is a text field to introduce a url related to the mapping, for instance, http://socialisingaroundmedia.com
5. Repository. It is a combo box field, where the user should select the “SAM” option
6. Read Access. It is a text field with several valid values, the user should keep this field empty

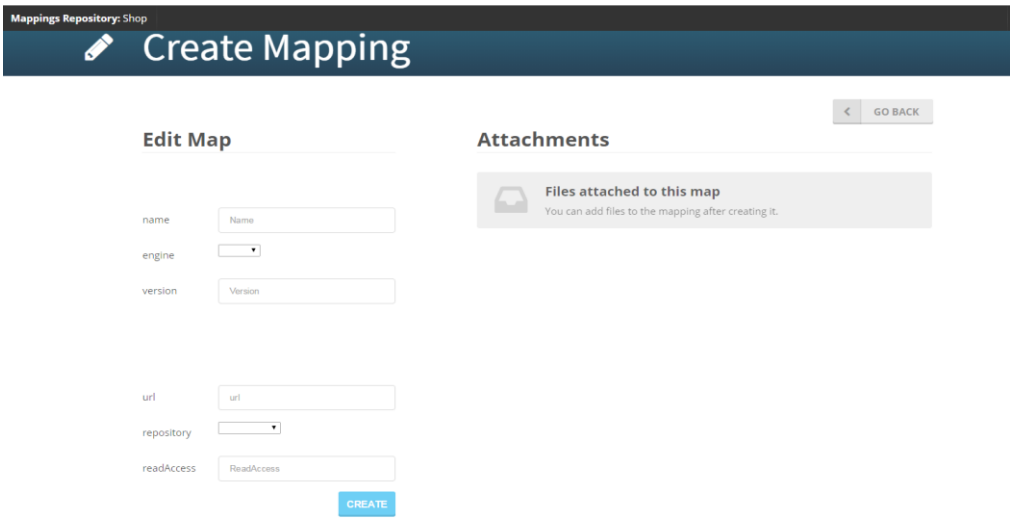

Figure 38: Mapping Repository - Create Mapping

3.4.2.3 Edit Mapping

Once the mapping is created, it is possible to attach files related to the mapping such as the xml files, LPO files or SNF file. The following information should be introduced:

1. Type of the file
2. Short description of the file
3. Choose the file to upload

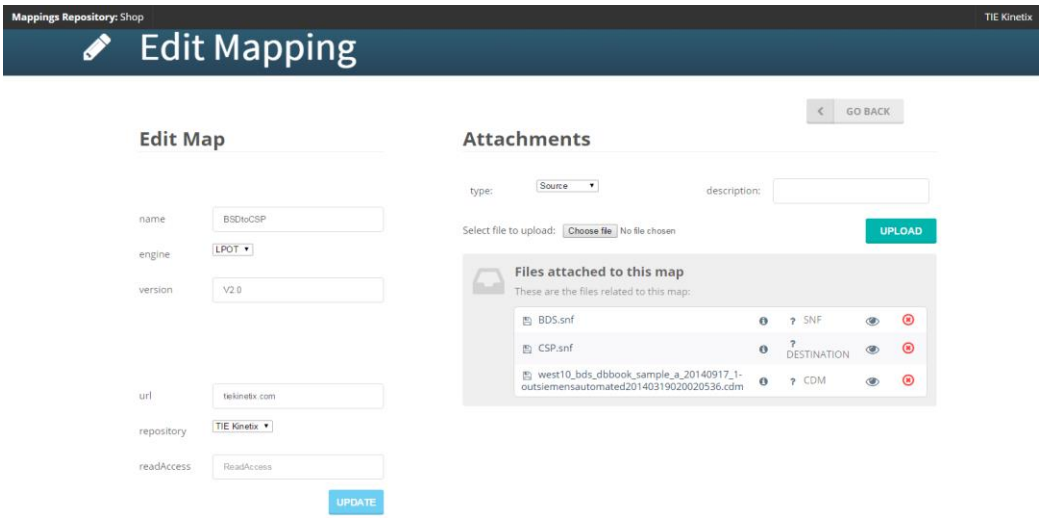

Figure 39: Mapping Repository - Edit Mapping

3.5 Limitations and Further developments

Due to the fact that this is the 1st Content Gateways prototype, there are still several limitations. The most important are the following:

- Semantic Integrator Editor is not connected with the Mapping Repository and Cloud Storage
- The Mapping Repository hasn't implemented the connection with the Identity and Security Services component for data authorisation, since this component is under development. The first prototype of T4.4 Distributed Identity, Trust and Security should be ready in M19 as specified in the DoW
- The communication with the Semantic Services component through Semantic Services controller has not been started

3.5.1 Prototype 2 Planned Tasks

Regarding the next steps in the Content Gateways component, Figure 40 and the following subsections summarise the tasks planned for the 2nd Prototype (to be delivered in M25).

Subcomponent	Task
Gateway Control	Develop the web services that are needed to allow the communication with the Syndicator, Interconnection Bus and Cloud Storage
Semantic Integrator Editor	• Integration with the Mapping Repository and the Cloud Storage • Mapping with the 1st SAM Asset Description (See D5.1.1 for further details on the SAM Asset description)
Mapping Repository	Integration with Identity and Security Services through the Marketplace
Web Data Extraction	• Research and implementation of alternative ways to carry out the crawling engine such as Google Search Appliance (GSA) • Develop the engine to provide the web data extraction functionalities
Semantic Services Controller	Integration with the Semantic Services component

Figure 40: Tasks Planning for the Content Gateways 2nd Prototype

3.6 Research Background

For the current prototype implementation and the overall approach the following papers have been taken into consideration:

Source	Subcomponent	Description
Semantic Interoperability for Technology-Enhanced Learning Platforms	Semantic Integrator Editor	This paper presents the results of the STASIS (www.stasis-project.net) project and applies it to the area of semantic interoperability within technology enhanced learning platforms.
STASIS - Creating an Eclipse Based Semantic Mapping Platform	Semantic Integrator Editor	This paper initially introduces the current schema mapping problem and outlines the limitations of existing solutions. The STASIS approach is then presented and contrasted with other semantic projects.
Simplifying e-Business Collaboration by providing a Semantic Mapping Platform	Semantic Integrator Editor	Within this paper the STASIS approach for creating a comprehensive application suite is introduced. It allows enterprises to simplify the mapping process.
Information Extraction Using Web Usage Mining, Web Scrapping and Semantic Annotation	Web Data Extraction	This paper aims to explain how to extract useful information from the web. It is the most significant issue of concern for the realisation of the semantic web. This may be achieved in several ways such as Web Usage Mining, Web Scrapping or Semantic Annotation.

Figure 41: Research Background Content Gateways

3.7 Target Performance

For this component the following key performance indicators (KPI) have been defined:

Topic	Description	Target KPI
File Import Time	The basic function of the Semantic Integrator Editor is to import files to be mapped. Thus, the time to load and import these files should be as minimal as possible.	• For XML files of 100Kb < 1 second • For XML files of 500Kb < 3 seconds • For XML files of 1Mb < 6 seconds

Figure 42: Target Performance Content Gateways

3.8 Summary

This section provides a description of the first prototype of the Content Gateways component developed in task T5.2 Content Gateways. The main outcome of this task is the software of the Content Gateways component. This prototype is the first of the three iterations planned for this component.

The most important goals reached during this 1st prototype have been:

- 1st version of the Semantic Editor Tool and progress in the mapping to transform BDS files into Syndicator format
- 1st version of the Mapping Repository
- Integration tasks in Semantic Services component services necessary to carry out the Content Gateways functionalities.
- 1st version of the Asset Import interfaces accessible through the Marketplace (See section 6.1.4.1.7)

And, the necessary requirements were presented for both users and developers in order to manage these goals.

The section 3.5 described the limitations of the current prototype such as the communication with Cloud Storage or the integration with Identity and Security services. It also described the next steps for the second version of the component, which should be delivered in M25.

4 Assets Aggregation and Composition

This section describes the software deliverable D5.3.1, which is the first prototype release of the SAM Assets Aggregation and Composition (Linker) component.

4.1 Scope and Relationship

The Linker component will be responsible for the Asset aggregation and composition for the SAM Platform. With this tool, the Content Providers will be able to enrich the Assets with additional information coming from other Assets, web or social network sources. The Linker will allow the Content Providers to define easily but also effectively all aspects of the new, composite Asset covering among others the aspects of asset time-line, audience types, delivery channels etc. This component will produce new Assets, annotated and enriched with information from various sources so as to create a rich 2nd Screen experience for the End Users. Figure 43 provides an overview of the Linker Component.

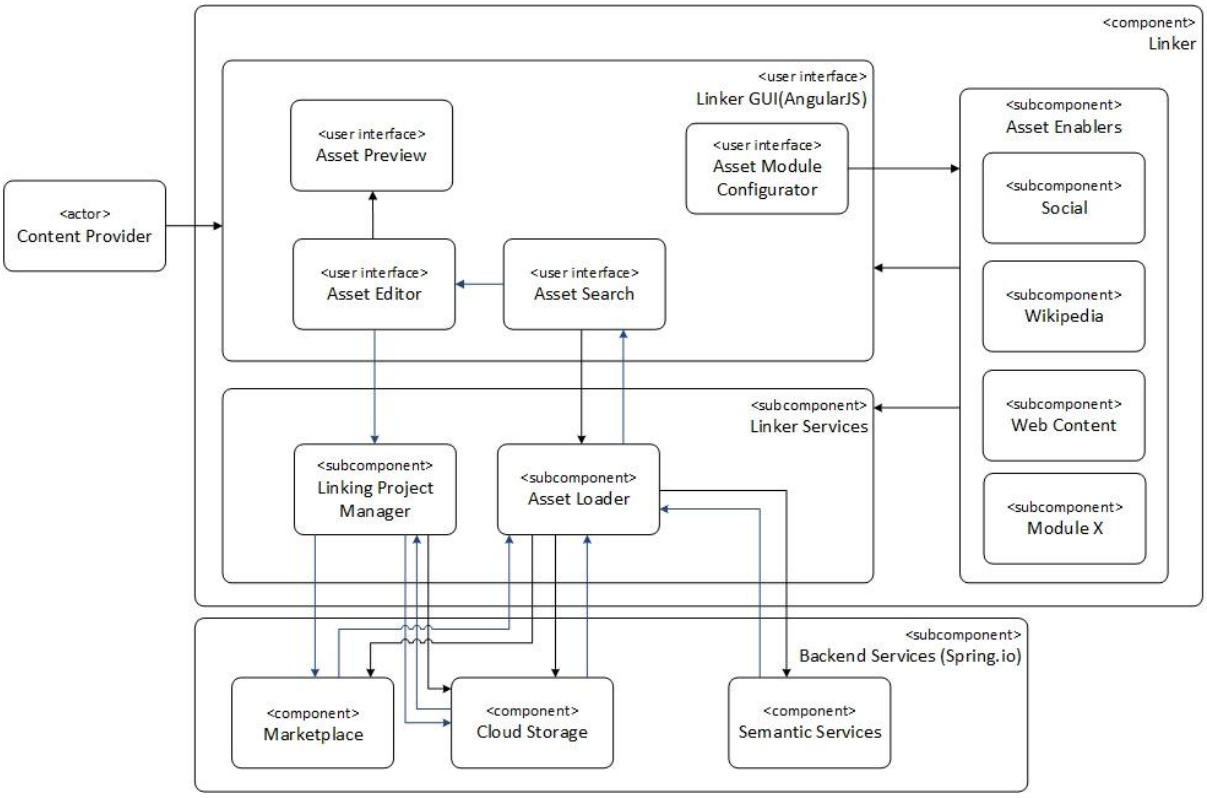

Figure 43: Linker Subcomponents and Relationships Between Them

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.11 (Architecture), D3.2.2 Section 4.12 (Functional Specification) or D3.3.1 Section 3.12 (Technical Specification).

As far as the backend functionality is concerned, the Linker component uses a RESTful interface in order to store or retrieve linking projects or Assets.

See Section B1.3.3.5 of the DoW for additional information on the effort distribution for this component in the lifespan of the project.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table (Figure 44).

Subcomponent	Task
Linker Multiscreen Timeline editing	Create a module in the UI to support the timeline editing of the Assets
Linker Asset Composition	The basic module for Asset Linking, can load connect and create Assets
Linker Project Manager	The project manager is a UI module for accessing, saving and creating new Linking Projects
Linker UI	Include SAM look and feel
Linker UI	Integrate the interface in the Marketplace
Linker Services	Define services for accessing, storing, deleting and creating new Assets.

Figure 44: Tasks Carried out for the First Prototype of T5.3

4.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users and developers, as well as software prerequisites for the installation of the Linker component.

4.2.1 For Users

Authorised users can use the online interface (HTML) to create or manage existing, linking projects. In the Linking projects the users can create, change or delete linked Assets. A screenshot of the current layout of this interface is shown in Figure 45.

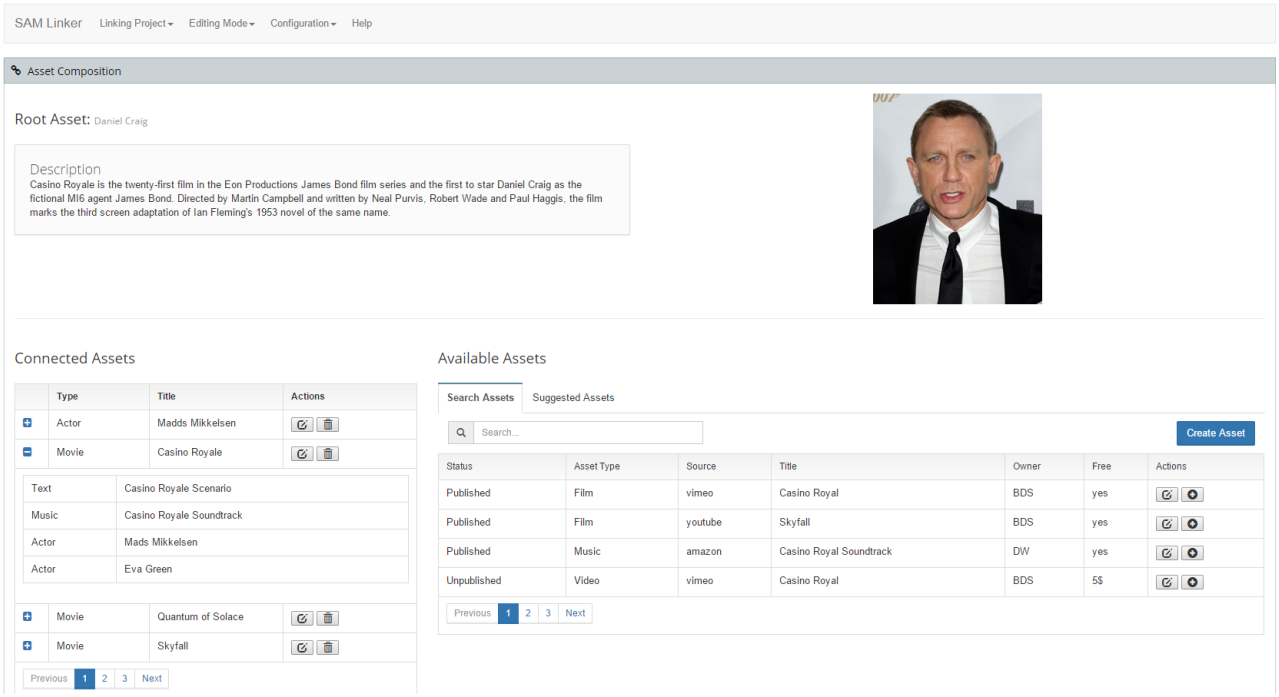

Figure 45: Linker Online Interface

4.2.2 For Developers

For developers who want to create their own Linker instance, both Java 7[8] and Apache Tomcat 8[9] are required. Apart of the user interface, the Linker component includes two RESTful Services that need to be deployed on the web application server: the Asset discovery and the profiler.

Additionally, developers can access the Asset discovery and the profiler services using the documentation demo page (see Section 4.4.6).

4.3 Installation (Deployment)

Currently the installation is carried out by the Jenkins Continuous Integration Server[10] provided by the SAM consortium. For the Linker component, a Jenkins integration project has been created and configured to build and deploy each subcomponent in Apache Tomcat 8.

In this first prototype, the Linker interface has been already integrated in the SAM Marketplace that will group all the functionalities related to the Marketplace of the SAM Platform.

4.4 Execution and Usage

This section describes the procedure to create a linked Asset, manipulate an existing one or previewing linked Assets. The two basic procedures are the Asset Composition and the Asset Timeline Linking. The first one allows the user to create linkage between Assets and the second one takes the linking a step further by allowing the linking of Assets for a

[8] http://www.java.com
[9] http://tomcat.apache.org/
[10] http://jenkins-ci.org/

specific time frame during a film or other video type Assets. The purpose of the preview is to help the users to check the correct alignment of the connected assets in order to avoid errors such as overlapping or wrong placement of Assets.

4.4.1 Access Interface: Linker UI

The Linker is embedded inside the SAM Marketplace, so users need to be registered in the Marketplace to access the Linker. In the Marketplace interface users can access the Linker from the left side vertical menu, as shown in Figure 46.

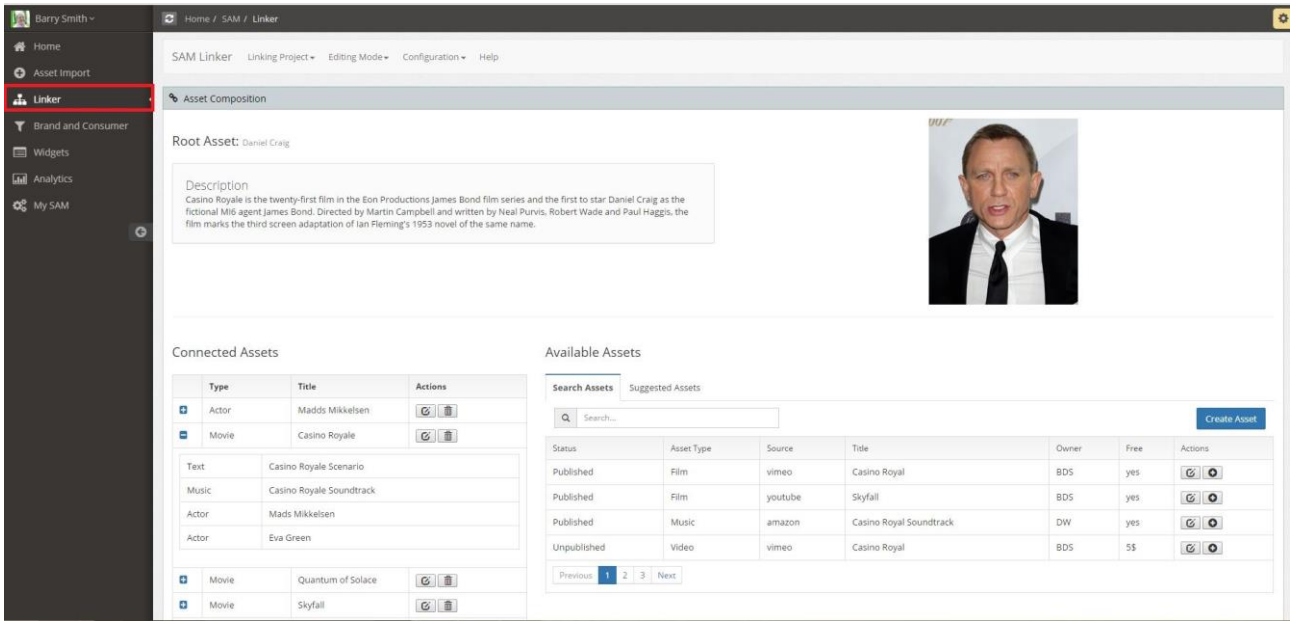

Figure 46: Linker Interface Integrated in SAM Marketplace

4.4.2 Linking Project: Linking Project Interface

The first step towards creating a Linking Project and consequently Link Assets is defining or accessing a Linking Project. In order to access, an existing Linking Project users should select from the dropdown list in the top menu (Figure 47) the Load Project field.

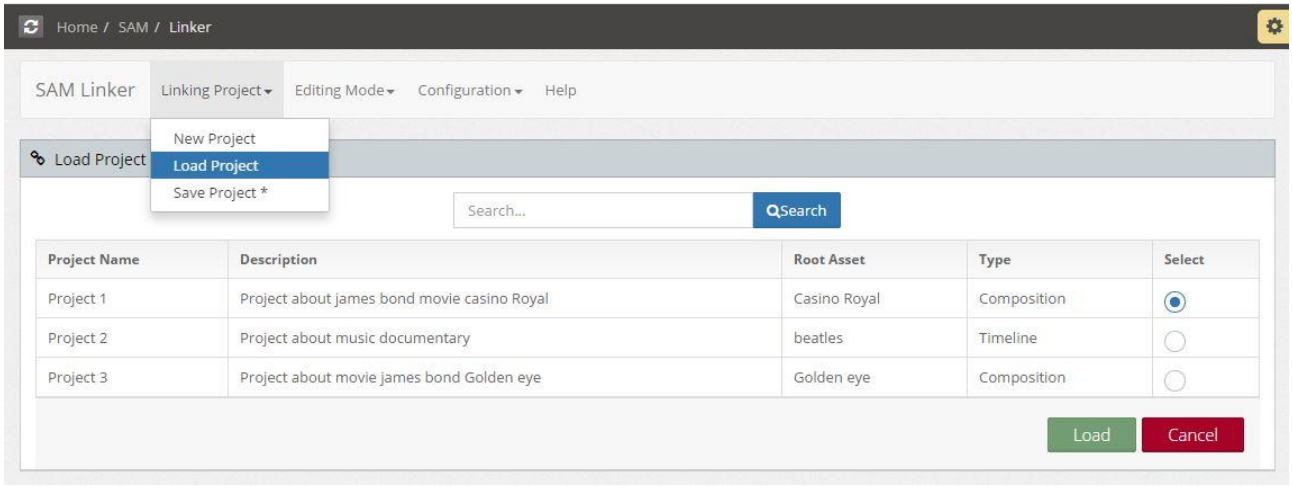

Figure 47: Linker Load Project UI

If users wish to create a new Linking Project, they should select the Create Project field. In the Create Project interface, users fill a form for the generic description of the Linking Project (title, description and project type) and then they assign the root Asset that they want to link with other Assets. The Root Asset assignment works as a starting point for the Linking (Figure 48).

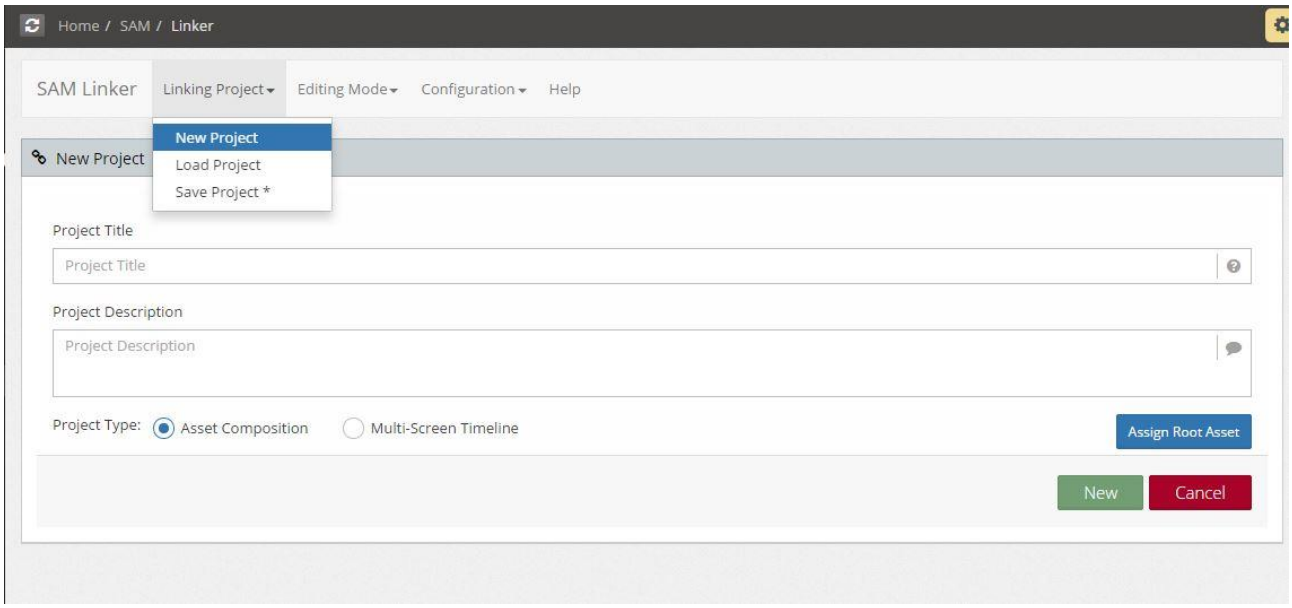

Figure 48: Linker New Project UI

4.4.3 Asset Linking: Asset Composition

Asset Composition is the core component of the UI for linking Assets. The first linking a user can create through this UI is a generic one. The generic linking is the association of the Root Asset with specific keywords and also giving a style for presenting these keywords (i.e. Default Theme). Going further, users can search for available Assets or create new ones in order to enrich the linking that is created.

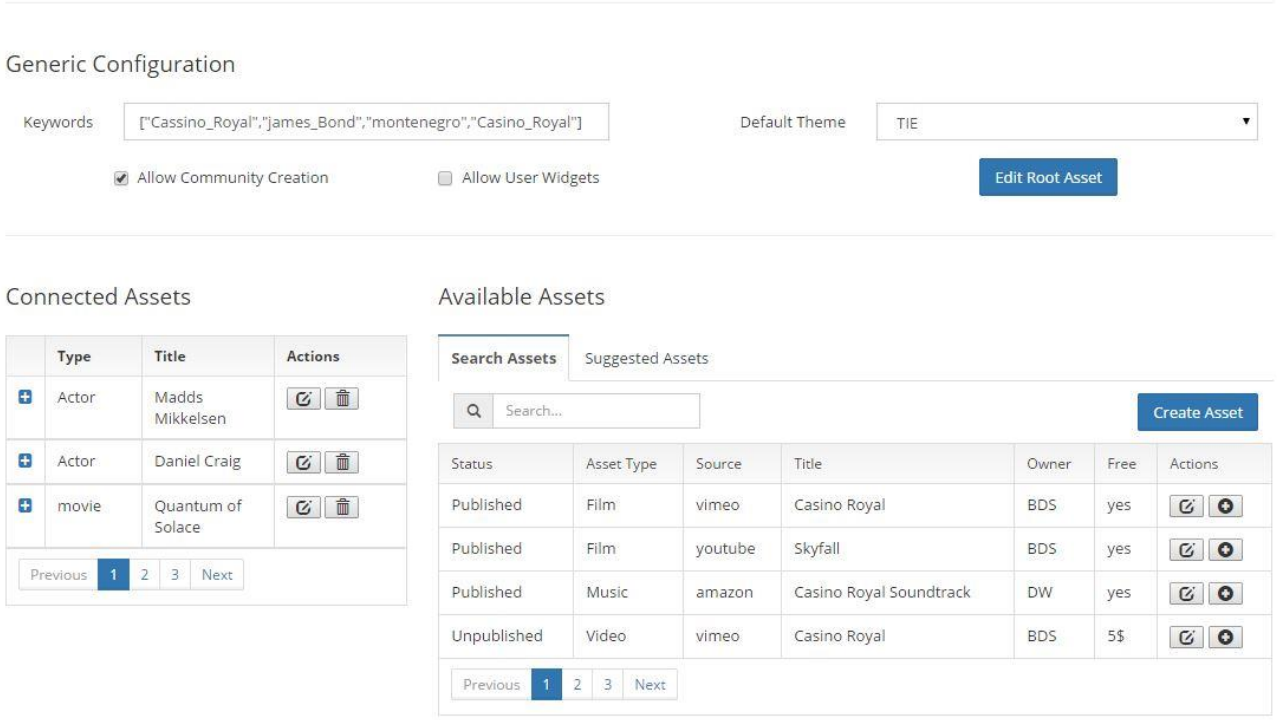

Figure 49: Asset Composition UI

By clicking on the “plus” icon in the actions field, users can link the selected asset with the Root Asset. In the pop up form that is shown users can define the specific way the Asset will be linked (Figure 50).

Figure 50: Asset Link modal

The functionality of the Linker allows the editing of Assets and also the creation of new Assets.

4.4.4 Asset Linking: Multi-Screen Timeline

The Multi-screen Timeline editor enriches the functionality of the Asset Composition by adding some additional features for the Linking. More specifically the users can Link Assets for specific periods of time (Figure 51), given that the Root Asset is a movie or an audio file.

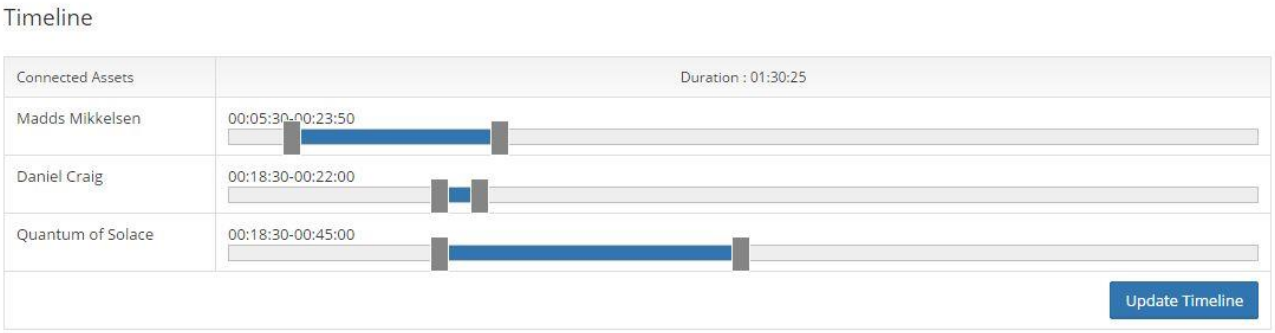

Figure 51: Linker Time-Line Visual Editor

As it is shown in Figure 51, the connected Assets are displayed in an array and users can change the starting and the finishing time of each Asset by adjusting the slider.

4.4.5 Additional Functionality

The purpose of the Linker is the linking of Assets. Nevertheless there is a need for extra functionality regarding the Asset manipulation. In the Linker UI users can also access the Assets and either create new ones or update existing ones. This functionality regards the Asset Profiler and it is essential to embed it inside the Linker in order to make the Linker more agile and user friendly.

4.4.6 Backend Functionality: Asset Discovery

For each Linker functionality there is a service that connects it with the Backend. Asset Discovery is the Rest API that enfolds all the services that Linker uses.

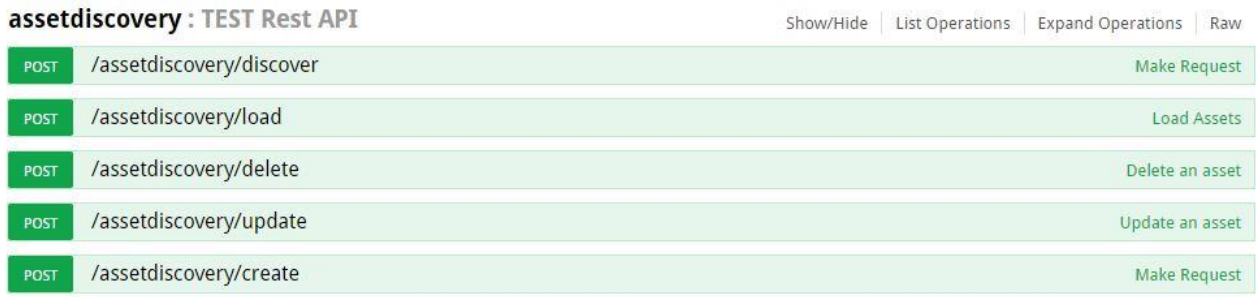

Figure 52: Rest API with the Linker Services

The first service is responsible for discovering and retrieving Assets. This service does not perform a simple search procedure (searching for a specific name). In order to retrieve Assets, this service uses as criteria the keywords that describe the Assets and also the context of the Asset. As shown in Figure 53.

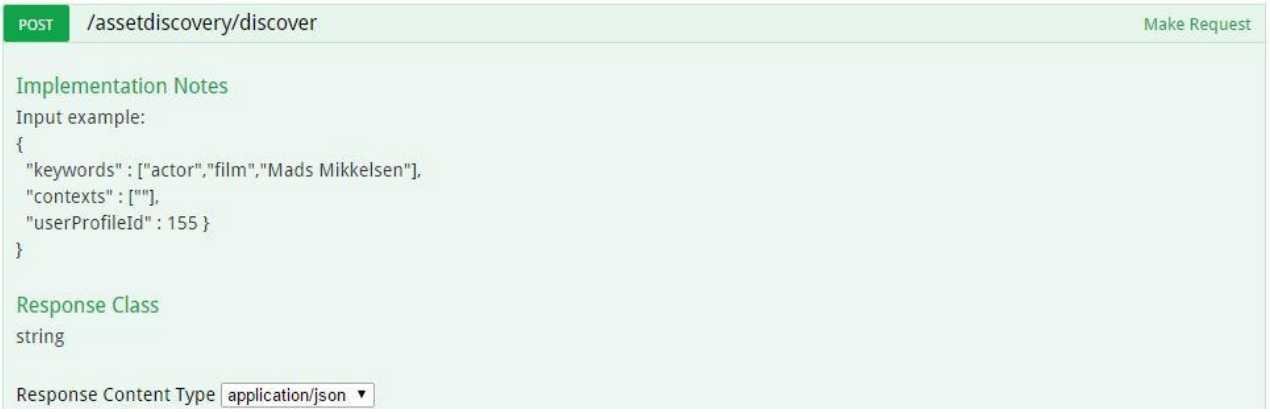

Figure 53: Asset Discovery Service

The second service is responsible for loading the Assets and serving them to the UI. This service populates with Assets the “Available Assets” module of the UI. The key feature of this service is to eliminate some of the Assets that are going to be served. If this service was to make just a simple request for loading the Assets, every Asset in the SAM database would be loaded. But this is not possible due to the size of the database and also practical user considerations. Thus, in order to eliminate the overload of the request and loading only the essential Assets, the “load” service enables filters for the requests it performs as it is shown in Figure 54.

Figure 54: Load Asset Service

The “delete” service is a simple service for deleting Linked Assets or simple Assets. Only the unique identifier of an Asset (Asset Id) is required to call this service (Figure 55).

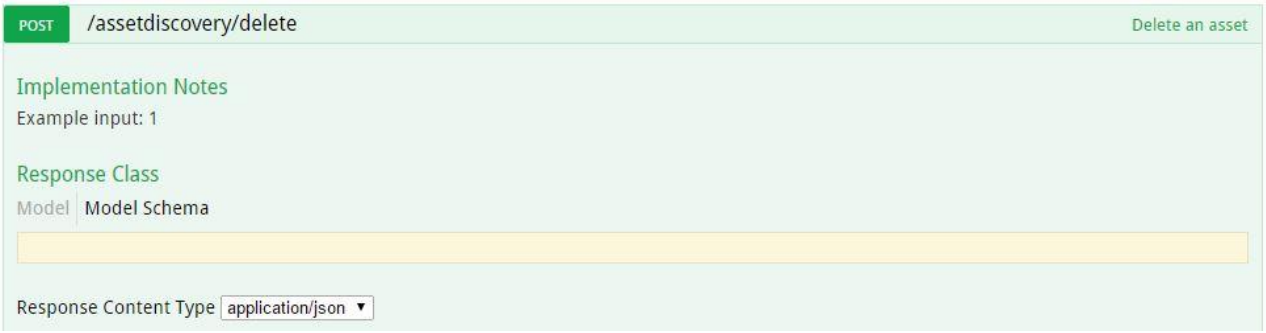

Figure 55: Delete Service for the Linker

The “update” service is also a simple service for updating the Assets or Linked Assets. The input of this service is the Asset itself with the updated fields (Figure 56).

Figure 56: Update Asset Service

In order to store new Assets a service for storing them is needed. This service takes all the information of the Asset in a structured way and posts them to the Cloud Storage (Figure 57).

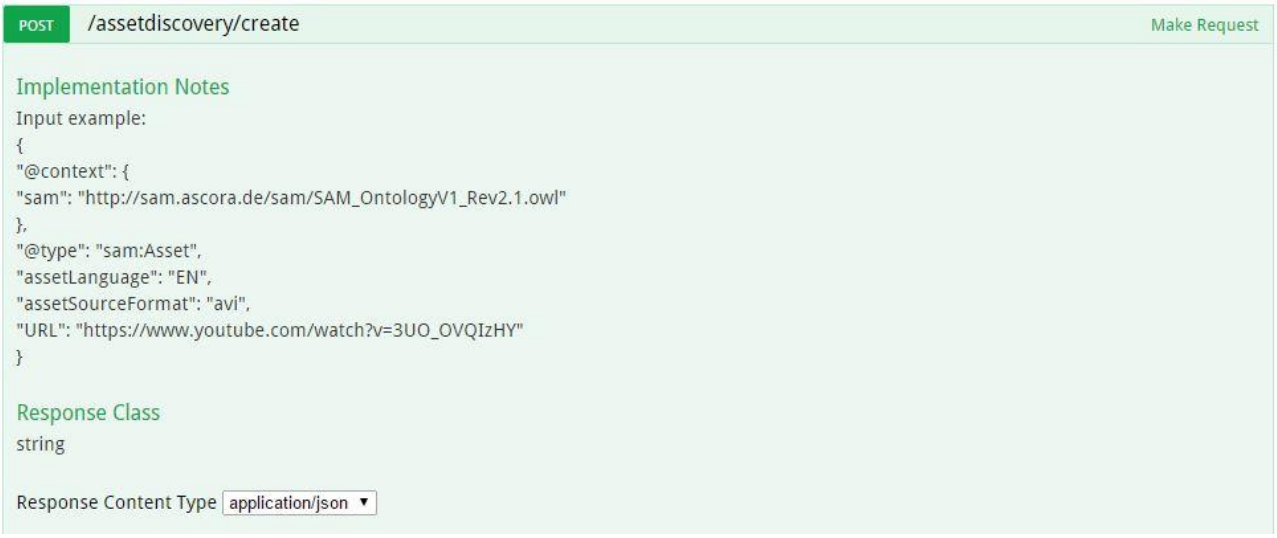

Figure 57: Create Asset Service

4.5 Limitations and Further Developments

4.5.1 Limitations

As first limitation, this first prototype will not support a fully functional Project Manager due to the Linking Projects should be stored in the Cloud Storage (see Section 3 in D4.9.1) and it is not still ready to manage Linking Projects.

Another limitation regards Module enabling functionality. At this stage the Linker cannot support multiple types of Assets. The Module enabling functionality combines the modularisation of web sources such as Wikipedia, YouTube and others. This functionality has many dependencies with other SAM Tasks, such as Marketplace or content

Presentation. Besides, it is one of the advanced features of the Linker and it is better to develop it in a more stable and advanced Linker prototype.

The Preview functionality is also an advanced feature of the Linker. The purpose of the Preview is to present in a more simple way the Linking of the Assets. All the Linker's components should be fully developed and functional in order to create the Preview component.

The last functionality that is not implemented yet in the Linker is the suggestion of Assets for Linking. This is an advanced backend functionality that not only retrieves Assets but also creates suggestions depending on the context keywords and the semantic representation of the Assets.

4.5.2 Prototype 2 Planned Tasks

The next steps regarding the Linker are eliminating the limitation described in section 4.5.1. The main efforts will be focused on fully linking the Assets through the UI, which means enabling semantic annotation, to enable modules and also create Social Media Linking. Enhancing the functionality of the RESTful services (Asset recommendations, advanced search) is also a planned advancement of the Linker.

Subcomponent	Task
Linker Project Manager	Enable the storage and retrieval of Linking Projects
Linker Module Enabler	Enable the creation and linking of modules through the UI
Linker Preview	Enhance the Linker with the Preview functionality
Linker Rest Services	Create services for Asset recommendation and advance search

Figure 58: Planned Tasks for the Linker's Second Prototype

4.6 Research Background

For the current prototype implementation and the overall approach the following related research work was taken into consideration:

Source	Subcomponent	Description
LANTHALER, Markus; GÜTL, Christian. On using JSON-LD to create evolvable RESTful services. In: Proceedings of the Third International Workshop on RESTful Design. ACM, 2012. p. 25-32.	Linker RESTful service	This paper considers developing the services of the linker.
SEVERANCE, Charles. Discovering javascript object notation. Computer, 2012, 4: 6-8.	Linker UI	This paper is considered as a guide for the linking of data (Assets) through the UI.
DARWIN, Peter Bacon; KOZLOWSKI, Pawel. AngularJS web application development. Packt Publ., 2013.	Linker UI	The Linker is developed using AngulaJS. So this paper can give useful information and guidelines for creating the UI of the Linker.

Figure 59: Research Background of the First Prototype of the Linker

4.7 Target Performance

For this component, the following key performance indicators (KPI) have been defined:

Topic	Description	Target KPI
Ease of use	The ease of use is an important aspect of the Linker. Users use the Linker UI in order to create multiple tuples of linkage between Assets. A user-friendly environment is a necessity in order to take advantage of all the functionality the Linker can offer.	Based on a short feedback questionnaire from the overall component owner the following metrics are taken under consideration. • Completion rate • Usability problems • Task time • Errors • Page Views/Clicks
Availability	The Linker‘s services should have high availability as multiple components may request Assets or Linking Projects at the same time.	Upon the second prototype, when integration with other components will have taken place, the availability should be at least 95%.
Precision and Recall	The Linker’s services not only retrieve Assets from the databases but also make suggestions, recommendations and also support advanced search. The precision and recall measurements are very strong indicators for examining how well the services work.	After analysing multiple queries, precision should be at least 80% and recall 90%.
Accessibility	The Linker should work in the most important browsers and it should be well formed XHTML markup and CSS markup.	Linker should work in Chrome, Opera, Safari, IE and FireFox. Pass the W3C validations.

Figure 60: Key Performance Indicators for Linker Component

4.8 Summary

This section describes the first prototype of the Linker (Asset Aggregation and Composition). More specifically, it describes the scope and relationship of this component with the Content Syndication component. Moreover, the way users or developers can access this component are described and also an analytic execution and usage guide is defined. Additional information such as limitations and further development, as well as research background and target performance is also provided in this section.

5 Brand and Consumer Protection

This section describes the software deliverable D5.4.2, which is the second prototype release of the Brand and Consumer Protection component.

5.1 Scope and Relationship

The BCP component is responsible for addressing aspects related to Asset content by applying protection filters in order to avoid unexpected content affecting both brands and End Users integrity. Figure 61 shows the different subcomponents of the BCP, the logical connections that have been established between them and the relations with other components and actors in the SAM platform.

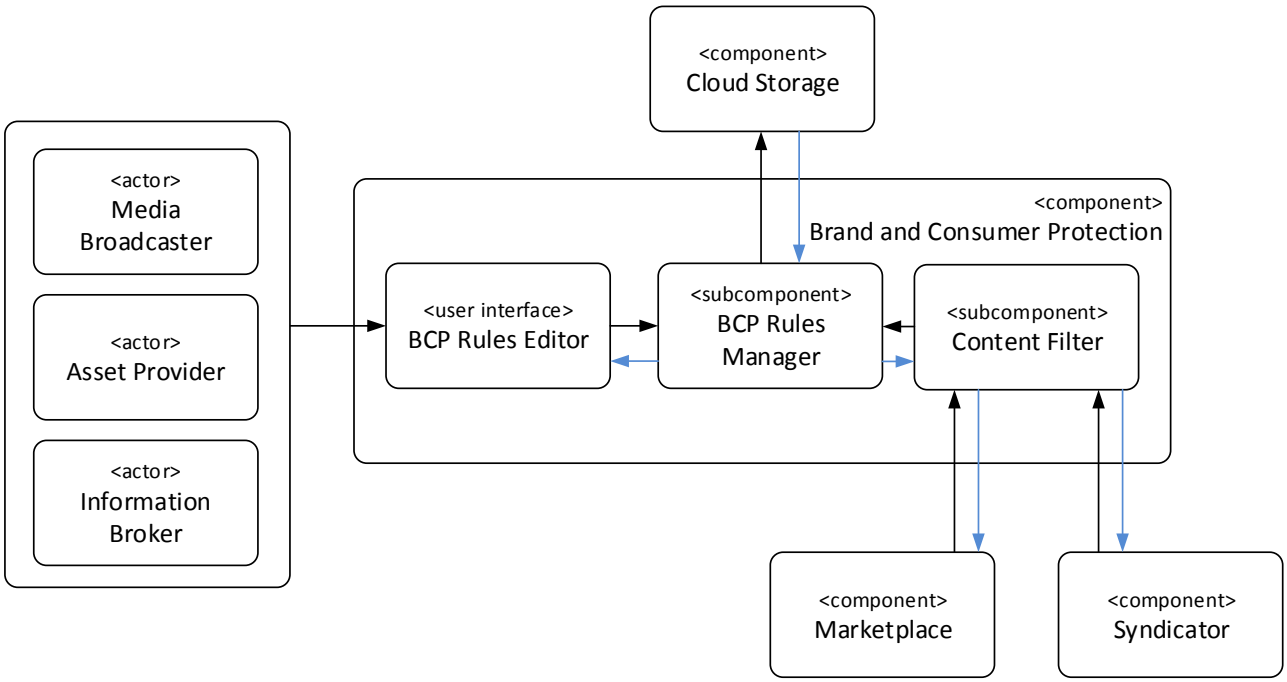

Figure 61: BCP Subcomponents and their Relationships

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.8 (Architecture), D3.2.2 Section 4.9 (Functional Specification) or D3.3.1 Section 3.9 (Technical Specification).

The main efforts in this second prototype have focused on the improvement of the rules format to allow the definition of complex rules combining logical operators explicitly (AND/OR), along with the connection to the Cloud Storage to store and retrieve rules. A summary of the tasks carried out for each subcomponent on this prototype is shown in Figure 62.

Subcomponent	Task
BCP Rules Manager	Update the JSON format for complex rules explicitly specifying logical operators.
BCP Rules Manager	Functions to create, read, update and delete rules from the Cloud Storage.
Content Filter	Update the evaluation of rules to integrate the new complex rules format.
BCP Rules Editor	Create, read, update and delete functionalities following the new complex rules format.

BCP Rules Editor	Differentiate fields belonging to Asset Structure or User Profile.

Figure 62: Tasks Carried Out for the Second Prototype of the BCP

5.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users and developers as well as software prerequisites for the installation of the BCP component.

5.2.1 For Users

Registered users (Protection Managers) can use the HTML online interface of the BCP Rules Editor to create, edit and delete rules that are used to filter and protect brands and End Users in SAM. A screenshot of the current layout of this interface is shown in Figure 63.

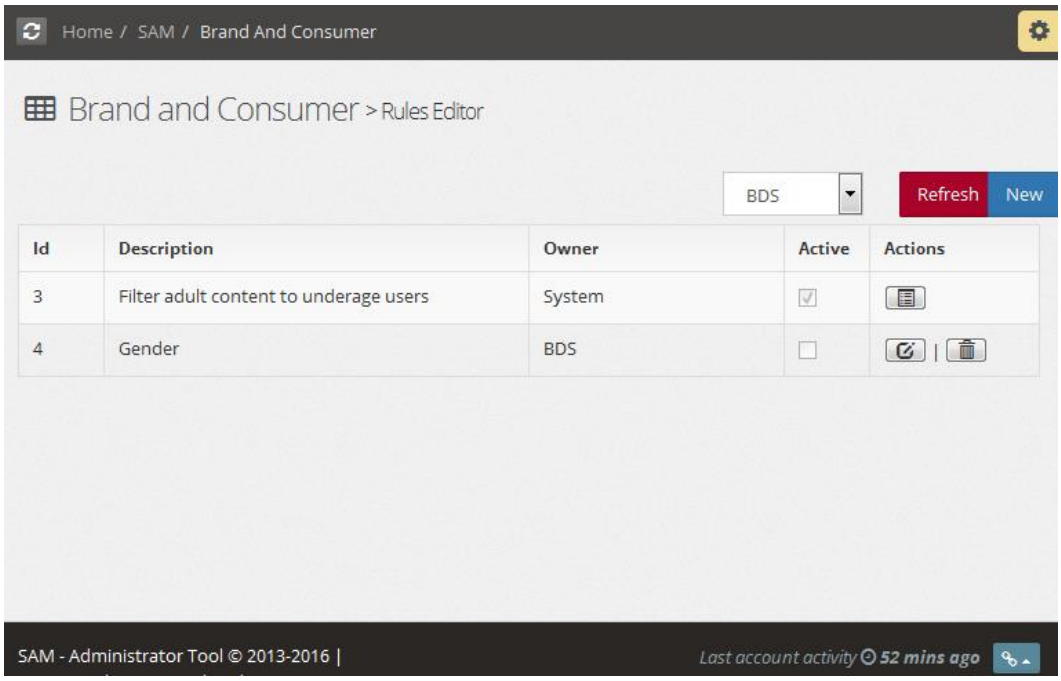

Figure 63: BCP Rules Editor Online Interface

The users of the BCP component (Protection Managers) only require a web browser to access the BCP Rules Editor interface.

5.2.2 For Developers

For developers who want to create their own BCP instance, both Java 7[11] and Apache Tomcat 8[12] are required. Besides the user interface, the BCP component includes two RESTful Services that need to be deployed on the web application server: the BCP Rules Manager and the Content Filter.

Additionally, developers can access the BCP Rules Manager and Content Filter services using the documentation demo page (see Section 5.4.2).

[11] http://www.java.com
[12] http://tomcat.apache.org/

5.3 Installation (Deployment)

Currently the installation is carried out by the Jenkins Continuous Integration Server[13] provided by the SAM consortium. For the BCP component, a Jenkins integration project has been created and configured to build and deploy each subcomponent in Apache Tomcat 8. The set up steps are the following:[14]

1. Download the project SAM Git repository.
2. Edit the file BCPApplication.java[15] and modify the following line as:
   `$swaggerConfig.setBasePath("http://[serverURL:portNumber]/[projectName]/api/v1"),`
   where `serverURL:portNumber` corresponds to the URL of the Tomcat web and its port number, and `projectName` refers to the name of the project.
3. Execute the following command-line:
   `$[root_directory_web_path] ant`,
   where `root_directory_web_path` refers the path of the web project's root directory (`brand-and-consumer-protection-web`).
4. The previous step will produce inside the `dist` directory a file named `[projectName].war`, which will be assigned the same name of the project. This file will constitute the Web Application Archive of the current web project.
5. Finally, to deploy the war file in the web server application, it is necessary to go through the administration console of the Tomcat server, select the option `deploy application` and then select the file `[projectName].war` created in the previous step.

In this prototype, the BCP Rules Editor interface has been already integrated both in the SAM Administrator Tool, which groups all the functionalities related to the administration of the SAM Platform, and in the SAM Marketplace, where functionalities regarding publish and access existing Assets are included.

5.4 Execution and Usage

This section introduces the required steps to create a rule that checks the state of an Asset, both in the BCP Rules Editor (Section 5.4.1) and in the BCP Rules Manager subcomponent (Section 5.4.2). Finally, the Content Filter subcomponent is presented in Section 5.4.3.

5.4.1 Creating a Rule (BCP Rules Editor)

In order to access the BCP Rules Editor, the user needs to be registered in the SAM platform and logged in the Administration Tool or the Marketplace. The "Brand and Consumer" option in the left side menu provides access to the BCP Rules Editor (see Figure 64).

[13] http://jenkins-ci.org/
[14] The requirements for setting up the web application are: Tomcat 8.0, Java 1.7 and Ant 1.7.1.
[15] This file can be found in: brand-and-consumer-protection-web/src/java/es/ua/sam/bcp/web/application

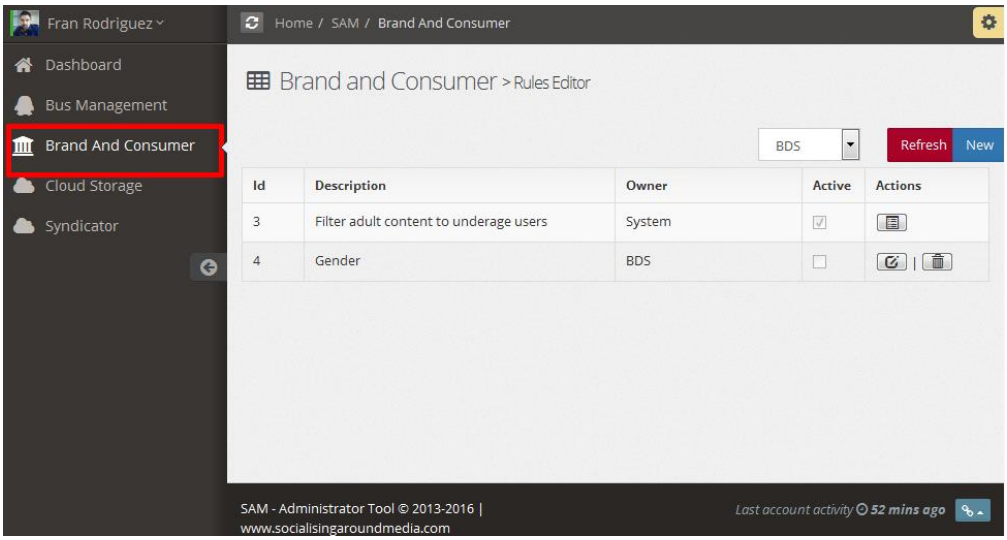

Figure 64: Rules Editor Interface Integrated in the SAM Administrator Tool

In the Rules Editor interface, the button “New” opens the “New Rule” dialog shown in Figure 65. In this window, users can add a description for the new rule, select the owner of the rule[16], activate it, and insert different filtering criteria.

With respect to filtering criteria, rules can contain different statements (new statements can be added by means of the “+Criterion” button shown in Figure 65) combined with logical operators (“+AND” and “+OR” buttons). First, all logical operators must be inserted, and then one or more statements can be added nested for each operator.

Figure 65: New Rule Dialog in the Rules Editor Interface

Each statement contains the following elements (see Figure 66):

- The first column represents the subject of the statement, indicating which End User or Asset feature is going to be evaluated in the rule. In this second prototype, these features are clearly differentiated to indicate which ones belong to the Asset Structure and which ones refer to the User Profile. Possible values in this column include “User Age” and “Rating”, indicating the age of the user and the rating of an Asset (i.e. indicating suitable ages for a film) respectively.

[16] In this second prototype, rules are not attached to a particular Protection Manager in the SAM platform, since profiling functionalities are not yet available. Thus, a dropdown menu allows selecting an owner for the sake of testing the component.

- The second column shows the conditions that can be applied to the selected field (e.g. “GREATERTHAN” and “EQUALS”)
- The last column represents the value with which the field is going to be compared by means of the condition selected. It can be a numeric value (e.g. “18” for the field “User Age”) or a string (e.g. “Movie” for the field “Asset Type”).

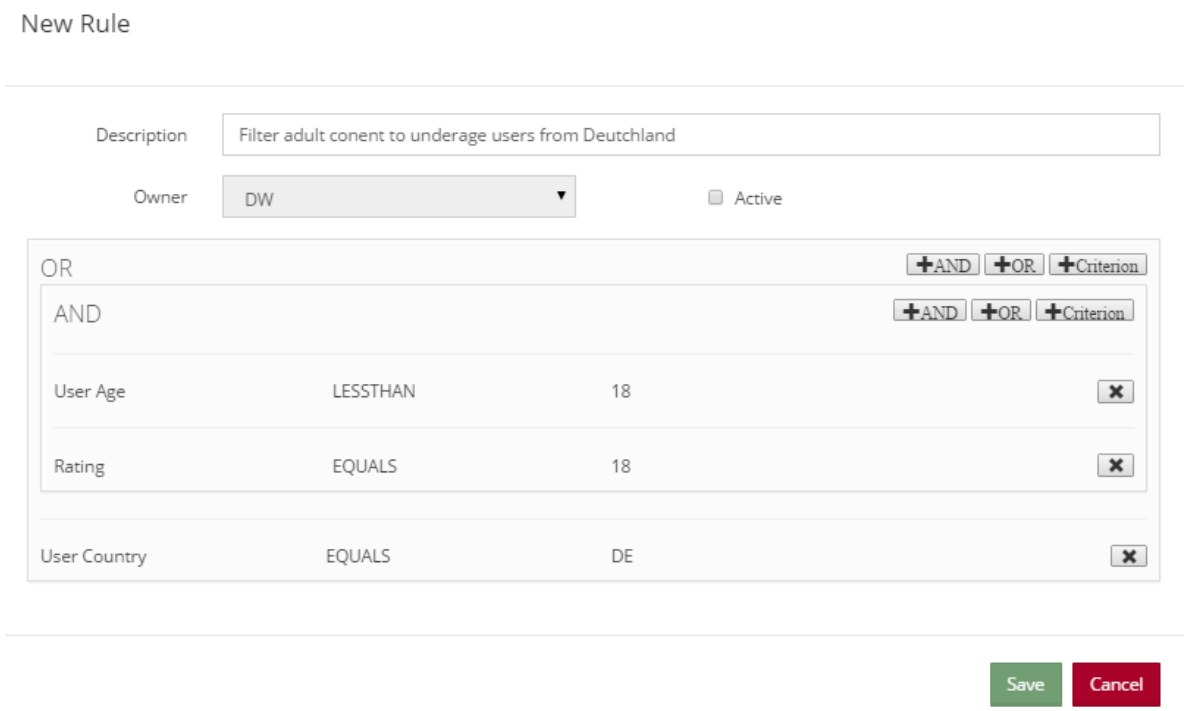

Figure 66: Example of a Complex Rule with Three Criteria and Two Logical Operators

Figure 66 shows a rule in the user interface composed of three criteria combined by two logical operators (an “OR” and an “AND”). The first statement establishes that the User Age is less than “18”. The second one indicates that the “Rating” of the Asset is “18” (suitable only for adults). The last one specifies that the “User Country” is “DE” (Germany). The first and the second criteria are connected using the “AND” logical operator, whereas the latter is combined to them using the “OR” logical operator. This rule could be described as “Filter Assets classified for adults to underage users from Germany”.

5.4.2 Creating a Rule (BCP Rules Manager)

The interface previously shown relies on the BCP Rules Manager subcomponent to manage rules. The documentation demo page of the BCP Rules Manager (see Figure 67) provides an interface where back-end functionalities corresponding to this subcomponent can be tested. This documentation exposes for each HTTP method at least one input example, its response codes, and the URL of the RESTful Services to try the subcomponent out.

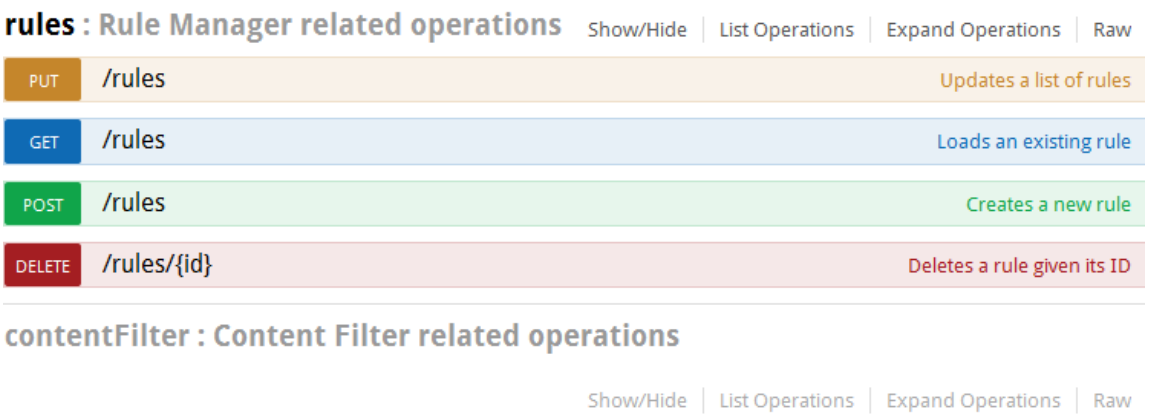

Figure 67: Documentation Demo Page for the BCP Rules Manager RESTful Service

In order to create a new rule, the method requires a JSON object containing the description of the rule, its owner, a flag indicating if it is active or not, and the filtering criteria. Figure 69 shows the rule “Filter Assets classified for adults to underage users from Germany” coded in JSON format.

```
{
  "description": "Filter adult content to underage users from Germany",
  "owner": "DW",
  "active": true,
  "id": 3,
  "criteria": {
    "type": "OR",
    "items": [
      {
        "type": "AND",
        "items": [
          {
            "type": "SIMPLE",
            "criterion": {
              "field": {
                "iname": "Rating",
                "jname": "asset.rating",
                "jtype": "int"
              },
              "condition": "GREATERTHANOREQUALS",
              "value": "18"
            }
          },
          {
            "type": "SIMPLE",
            "criterion": {
              "field": {
                "iname": "User Age",
                "jname": "user.age",
                "jtype": "int"
              },
              "condition": "LESSTHAN",
              "value": "18"
            }
          }
        ]
      },
      {
        "type": "SIMPLE",
        "criterion": {
          "field": {
            "iname": "User Country",
            "jname": "user.country",
            "jtype": "java.lang.String"
          },
          "condition": "EQUALS",
          "value": "DE"
        }
      }
    ]
  }
}
```


Figure 68: A Complex Rule Represented in JSON Format

Figure 70 shows a fragment of the documentation demo page for the “Create a new rule” operation. In the “Parameter” section, the body text box can be filled with a JSON example (as the one shown in Figure 69) and then check the result using the “Try it out!” button.

The section “Response Messages” provides information on the possible responses of the BCP Rules Manager when a rule is created.

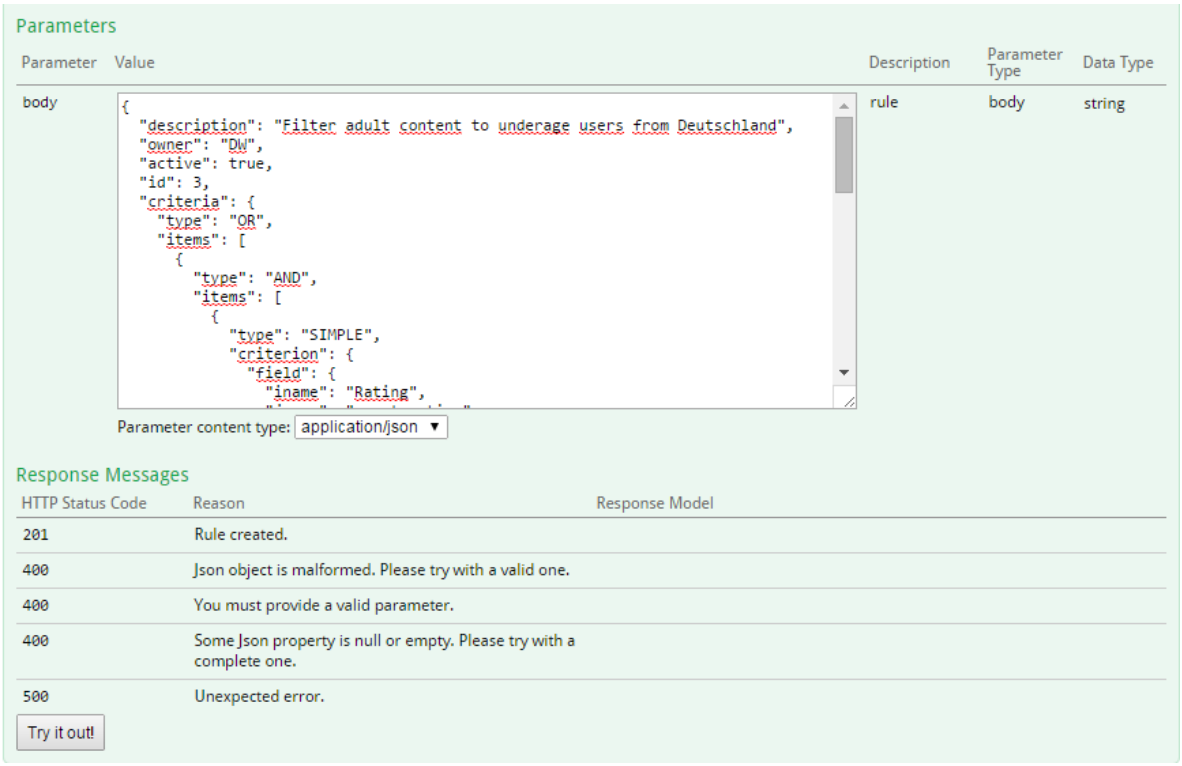

Figure 70: Creating a Rule in the Documentation Demo Page

5.4.3 Filtering an Asset

The Content Filter subcomponent filters Assets on demand given a target End User and a rule owner. The back-end functionalities of this subcomponent can be tested using the documentation demo page mentioned in the previous section (see Figure 71). This documentation exposes for this HTTP method an input example, its response codes, and the URL for the RESTful Service of this subcomponent.

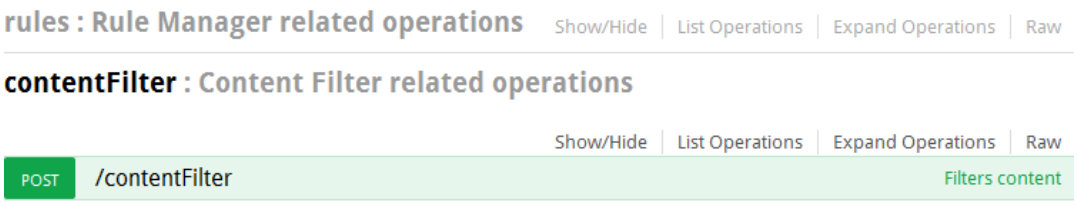

Figure 71: Documentation Demo Page for the Content Filter RESTful Service

Similarly to what was explained in the previous section, Figure 72 shows a fragment of the documentation demo page for the “Filters content” operation. The input of this component is a JSON object containing a list of Assets to filter, the profile of the target End User who wants to consume these Assets, and the owner of the Assets (whose rules must be applied). As a result, this operation returns the same list of Assets marked as approved or not according to the owner rules and the User Profile.

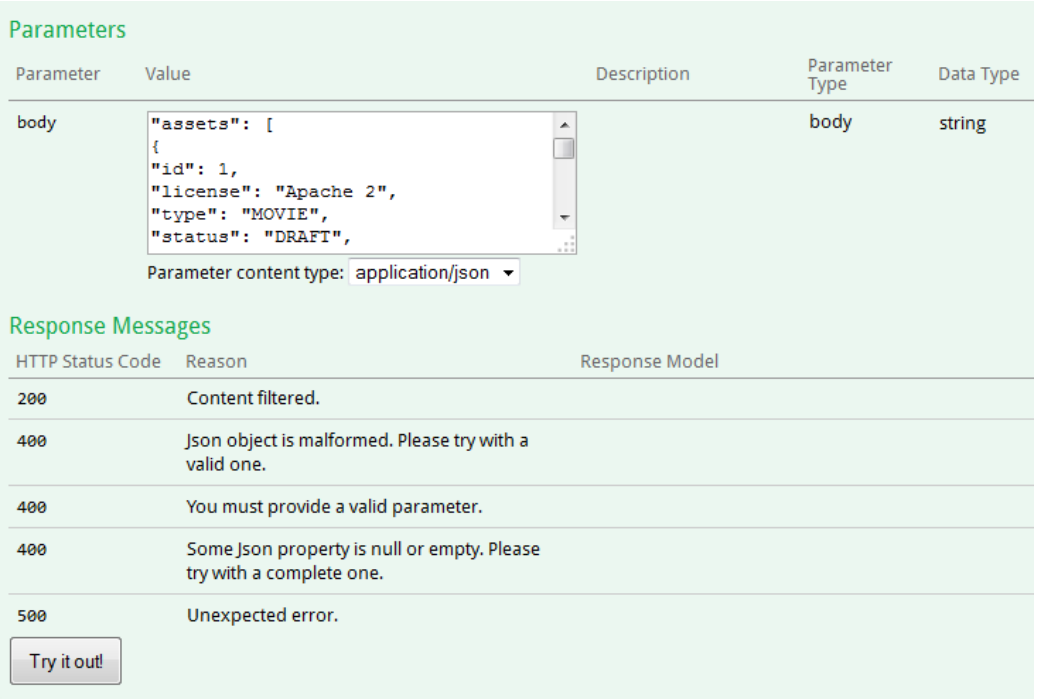

Figure 72: Filtering Assets from Content Filter RESTful Service

5.5 Limitations and Further Developments

5.5.1 Limitations

The main limitation of the second prototype is related to the set of possible fields that can be selected when creating statements to build the rules (see Section 5.4.1). The goal of BCP is to allow Protection Managers to define filtering criteria that take into account any possible feature of the Asset structure and User Profiles. The process of integrating this information is currently in progress, and will be finished when the next prototype is released.

In this second prototype a basic set of fields related to the user (Age, Language, Gender and Country) and the Asset (Type, Rating, Language, License Type and Status) has been defined. Also, some basic operations have been established, depending on whether the field requires a numeric (“EQUALS”, “NOTEQUALS”, “GREATERTHAN”, “GREATERTHANOREQUALS”, “LESSTHAN” and “LESSTHANOREQUALS”) or a string value (“EQUALS” and “NOTEQUALS”).

Another limitation of the prototype is the connection with the Cloud Storage in order to store and retrieve rules (see D4.1.1 Asset Storage & Information Management) for more information about the Cloud Storage). Currently, BCP communicates directly with the Cloud Storage, whereas in the next prototype it will be integrated in the platform by means of the Interconnection Bus (see D4.2.1 SAM Communication and Federation for more information about the Interconnection Bus).

To conclude, it should be noted that the set of rules defined by a Protection Manager are implicitly connected by a disjunction operator (OR). This means that when someone defines a set of rules, Assets are filtered to the users if any of these rules are true. For instance, if a Protection Manager defines two rules such as “Filter Assets rated for adults

only to underage users" and "Filter Assets for users that do not live in the UK", their Assets would be filtered for any user that accomplish any of these two criteria.

5.5.2 Prototype 3 Planned Tasks

Regarding the next steps in the BCP component, Figure 73 summarises the tasks planned for the third (and final) prototype, to be delivered in M25. The main effort will be focused on the integration of rules that take into account the content of an Asset and all the fields contained in the Asset structure and in the User Profile, along with the use of the Interconnection Bus for the communication with the Cloud Storage.

Subcomponent	Task
BCP Rules Manager BCP Rules Editor	Update the fields, conditions and possible values according to the final Asset structure and User Profile.
BCP Rules Manager BCP Rules Editor	Allow comparisons between fields in the rules (e.g. the field "Rating" of a film must be lower than the field "Age" of a user).
BCP Black List Manager	Add CRUD operations to manage the black/white list of users given an owner.
BCP Rules Manager Content Filter	Configure Brand and Consumer Interfaces in the Interconnection Bus.
BCP Rules Manager	Integration with the Cloud Storage using the Interconnection Bus.

Figure 73: Tasks Planning for the Third Prototype

5.6 Research Background

For the current prototype implementation and the overall approach the following related research work was taken into consideration:

Source	Subcomponent	Description
Norman Balabanian; Bradley Carlson (2001). Digital logic design principles. John Wiley. pp. 39–40. Chapter 2.	BCP Rules Manager	The Content Filter subcomponent filters Assets according to certain rules established by the Asset's owner. These rules have been defined using Boolean algebra, in which variables represent information about Assets or Users. These variables are combined in a rule with the operators conjunction (AND) and disjunction (OR).

Figure 74: Research Background for the BCP component

5.7 Target Performance

For this component, the following key performance indicators (KPI) have been defined:

Module	Description	KPI
Content Filter	Measure the performance of the Content Filter submodule	Number of Assets that can be filtered in a minute > 1000
Rules Manager	Measure the performance of the Rules Manager submodule	Number of rules that each CRUD operation can process in a minute > 700
Rules Editor	Measure the owner satisfaction with the user interface	Average of minutes spent creating or updating a rule < 3

Figure 75: Target Performance for the BCP

5.8 Summary

This section provides a description of the second prototype of the BCP component developed in task T5.4 (Brand and Consumer Protection). The main outcome of this task is the software of the BCP component. This prototype is the second of the three iterations planned.

The main goals reached during this second prototype have been:

- Design and development of a new format for defining complex rules, specifying explicitly logical operators
- Connection with the Cloud Storage to store and retrieve rules

For both front-end and back-end functionalities, the necessary requirements have been presented for users and developers to manage them.

The last section was dedicated to describing the limitations of the current prototype and the next steps considered for the third (and final) version of the component, which should be delivered in M25.

6 Assets Marketplace

This section describes the software deliverable D5.5.1, which is the first Prototype release of the SAM Marketplace component (Section 6.1) and Data API Services (Section 6.2) as part of the Syndicator component.

6.1 Marketplace

This section describes the SAM Marketplace first prototype functionalities

6.1.1 Scope and Relationship

The Marketplace is the area where Information Brokers and Content Providers can publish and access existing Assets. In this way, they can make them available to the End Users or to allow other Content Providers and/or Information Brokers to use/enrich/buy these Assets and potentially monetise them. The Marketplace will be the main User Interface for the business user roles such as the Content Providers, Information Brokers and 3rd party Software Providers. It will provide a unified interface to access the main editors in SAM (such as Linker, Ontology Editor, etc.). The different subcomponents involved in providing the marketplace functionalities are shown in Figure 76.

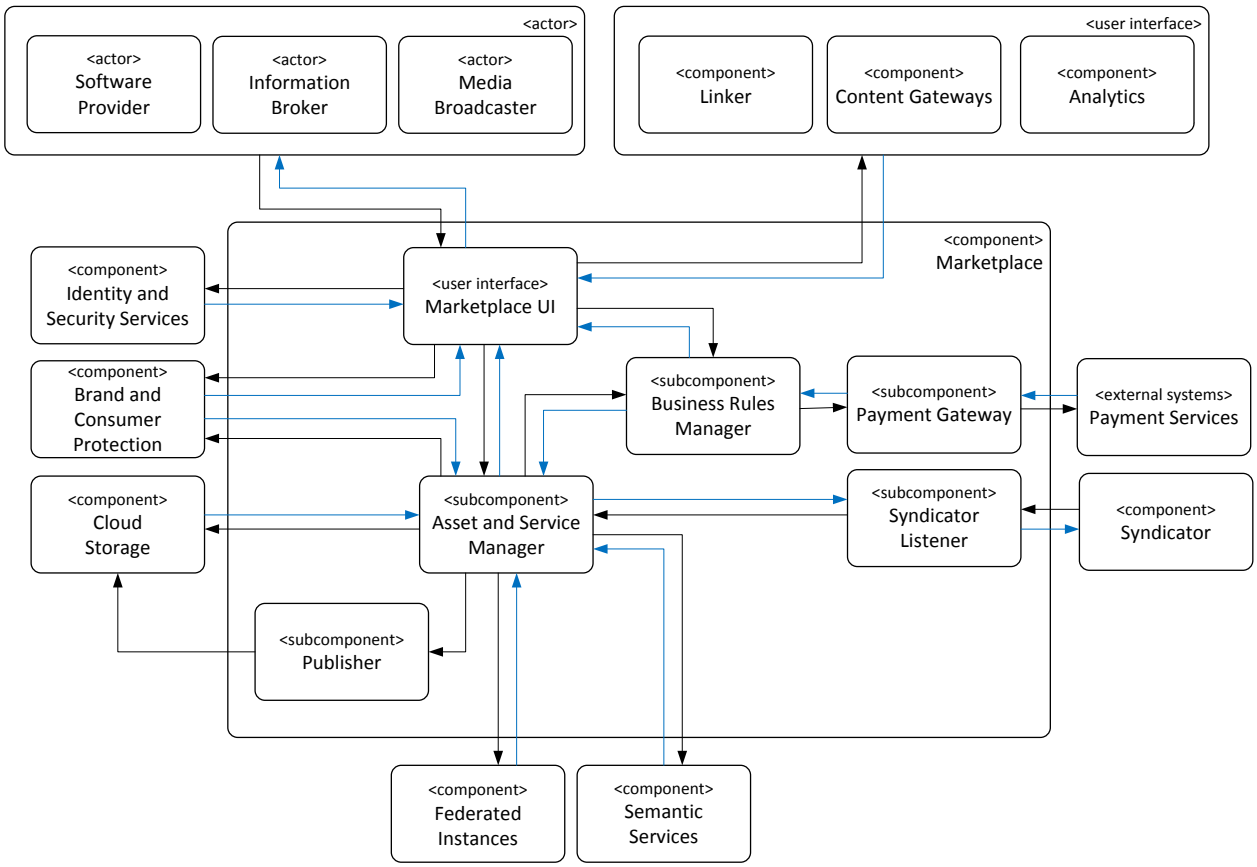

Figure 76: Marketplace

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.10 (Architecture), D3.2.2 Section 4.11 (Functional Specification) or D3.3.1 Section 3.11 (Technical Specification).

For this 1st Prototype, a first Marketplace User Interface template was agreed by the consortium and adapted. A draft version of this user interface was presented during the 1st Review Meeting embedding the different components mockups. The current version (1st Prototype) contains a more advanced design and features for the Marketplace functionalities:

- User Login and Create Account
- Dashboard. It contains different widgets to search Assets and view the Assets acquired by the End User
- Shopping Cart: 1st design of the shopping cart to buy assets and services
- My SAM: It provides the 1st design version for the Content Providers information area. It contains different widgets for managing the following information:
  - Company Details
  - Invoice History
  - Payment Gateways
  - Order Information
  - Sales Information

In addition to this, the integration with the different components has been started:

- View/Edit Asset (Semantic Services). See D4.9.1 Section 5 for further details
- Asset Importation (Content Gateways. See Section 3 for further details
- Link Asset (Linker). See Section 4 for further details
- Brand and Consumer. See Section 5 for further details
- Widgets (Multi-device representation). See D7.9.1 Section 3 for further details
- Analytics. See D6.9.1 Section 6 for further details

This template will be used for the rest of the components in order to keep the same style for the whole platform and provide a better user experience for the End User.

6.1.2 Requirements and Preparations

The Requirements and Preparations section describes the information needed to deal with the pilot, in terms of technical and non-technical requirements, software to be installed, basic knowledge, etc.

6.1.2.1 For Users

The user should use any of the most important browsers (Chrome, Firefox, Internet Explorer, Safari). The development and primary testing is being carried out on Chrome as agreed by the consortium.

6.1.2.2 For Developers

The Marketplace is built with AngularJS, thus is recommended to use an AngularJS Editor such as WebStorm[17] or Visual Studio[18] for the development.

[17] https://www.jetbrains.com/webstorm/download/
[18] https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx

6.1.3 Installation (Deployment)

The Marketplace installation is carried out by the Jenkins Continuous Integration Server[19] provided by the SAM consortium. A Jenkins integration project has been created and configured to build and deploy each subcomponent in Apache Tomcat 8.

6.1.4 Execution and Usage

The Marketplace is available through the web address provided in D5.5.1.

6.1.4.1 For Users

This section describes the steps required for a user to see the currently available Marketplace sections. Some of the areas are still not integrated, thus the related mockup is shown.

6.1.4.1.1 Login

Through this interface different End Users can enter their credentials. Once validated the End Users are logged into the system. The same interface also provides access to register new End Users or give a reminder of credentials.

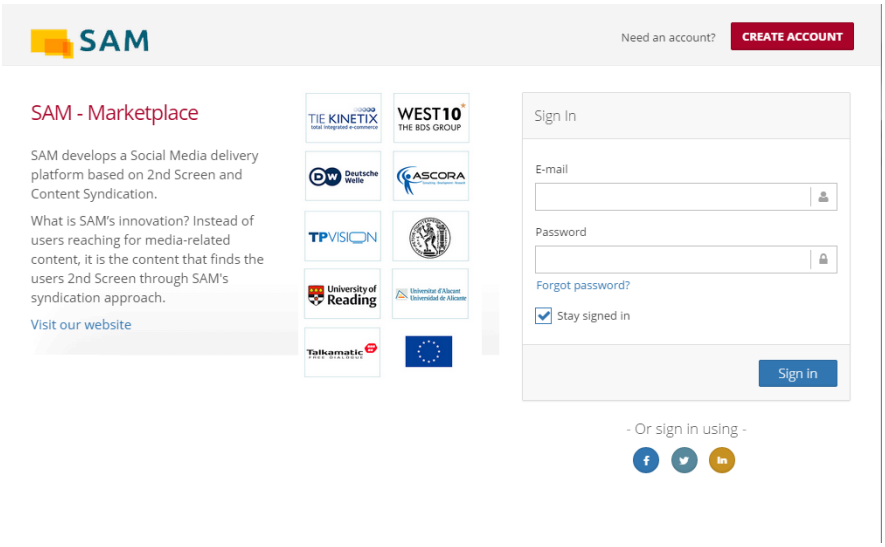

Figure 77: Marketplace - Login

6.1.4.1.2 Create Account

This user interface allows entering the necessary information in order to register a new End User. The basic information is: username, password, email, and company and industry sector.

[19] http://jenkins-ci.org/

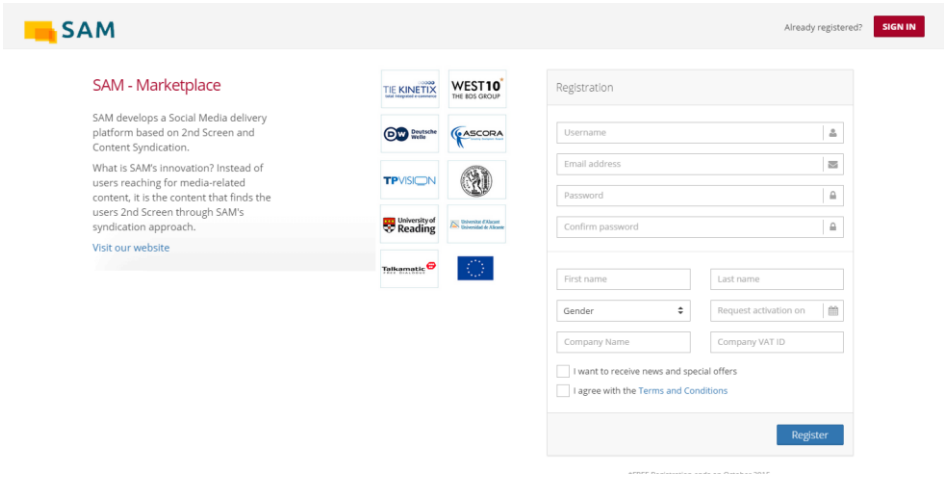

Figure 78: Marketplace – Create Account

6.1.4.1.3 Dashboard

The dashboard contains different widgets in order to provide different kinds of searches and to view/edit/link/sell SAM Assets.

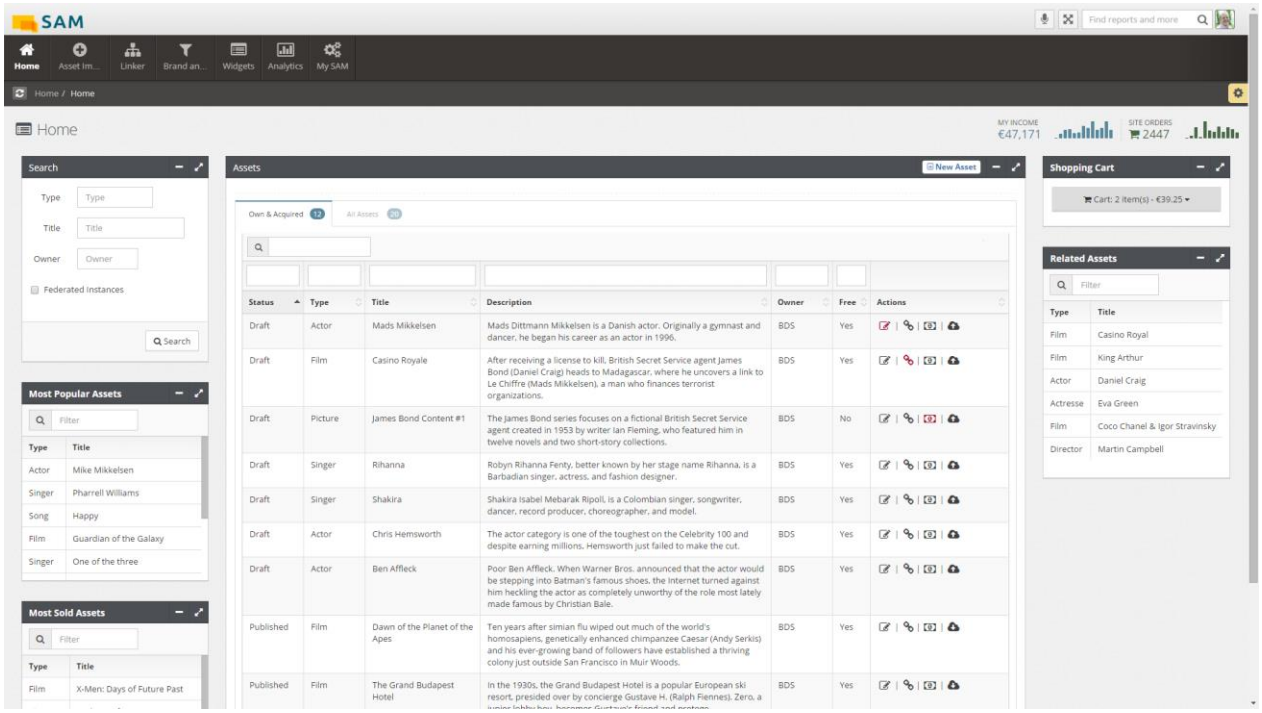

Figure 79: Marketplace – Dashboard

6.1.4.1.4 Shopping Cart

The End Users can buy Assets and services published in the SAM Marketplace. These will be shown in the tab “All Assets” in the Dashboard. The End User should click on the Asset shopping cart to add it to the final list.

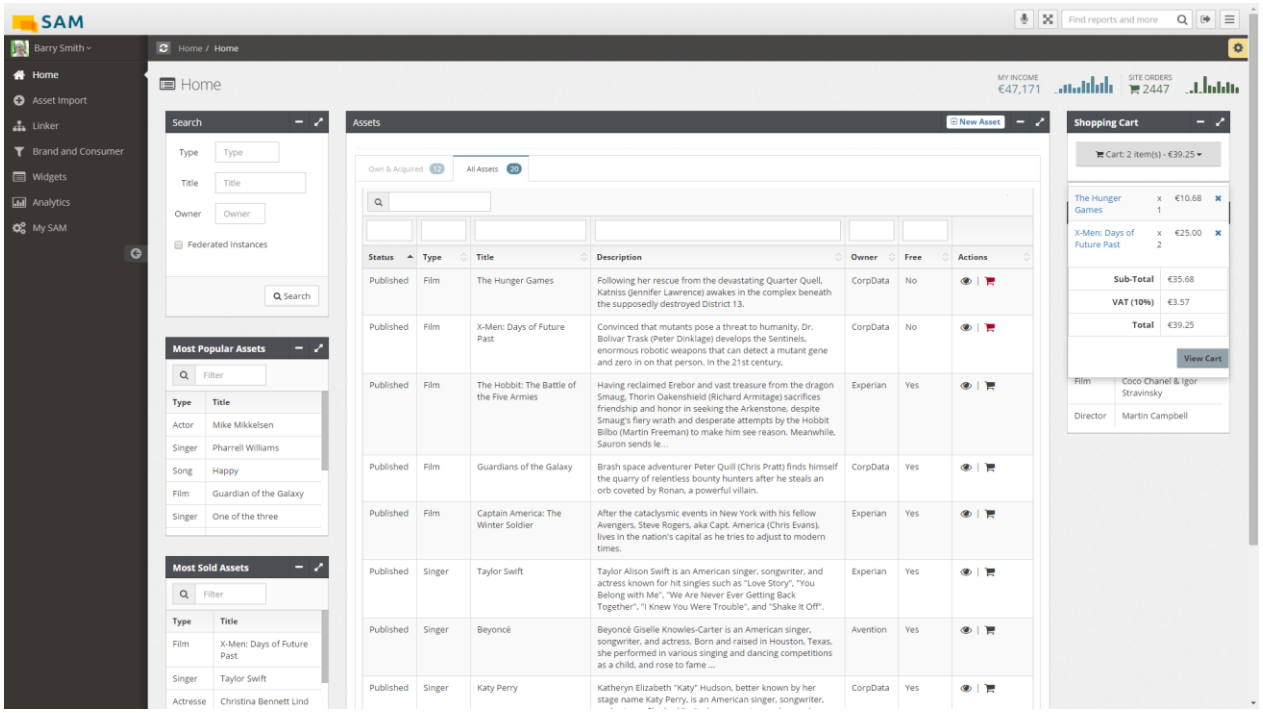

Figure 80: Marketplace – Shopping Cart

The details of the Shopping Cart will be shown by clicking on the View Cart button.

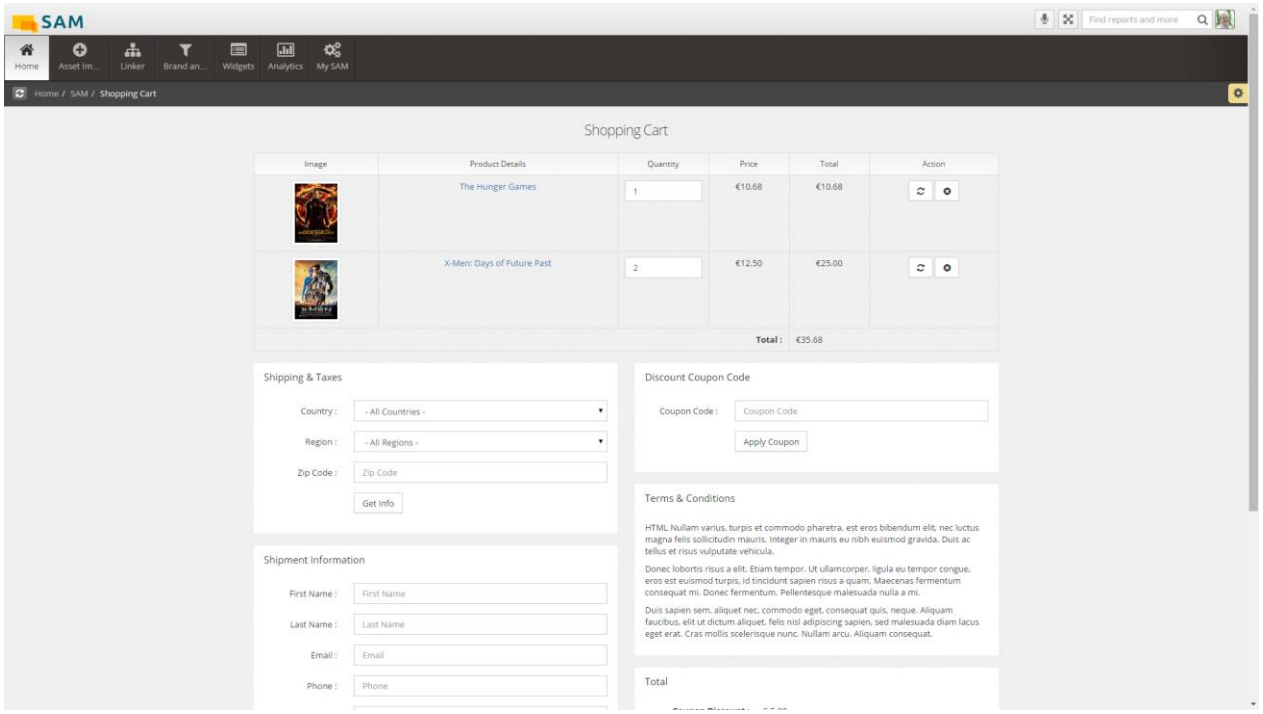

Figure 81: Marketplace – Shopping Cart Details

6.1.4.1.5 View/Edit Asset

This interface provides an area where the End User is able to enter the basic Asset features. There are boxes for setting restrictions (age constraints, validity time limitation), type, owner, source, license, logo, features and status of the Asset. In addition, a preview of the widget can be obtained at any time regardless of the selected tab.

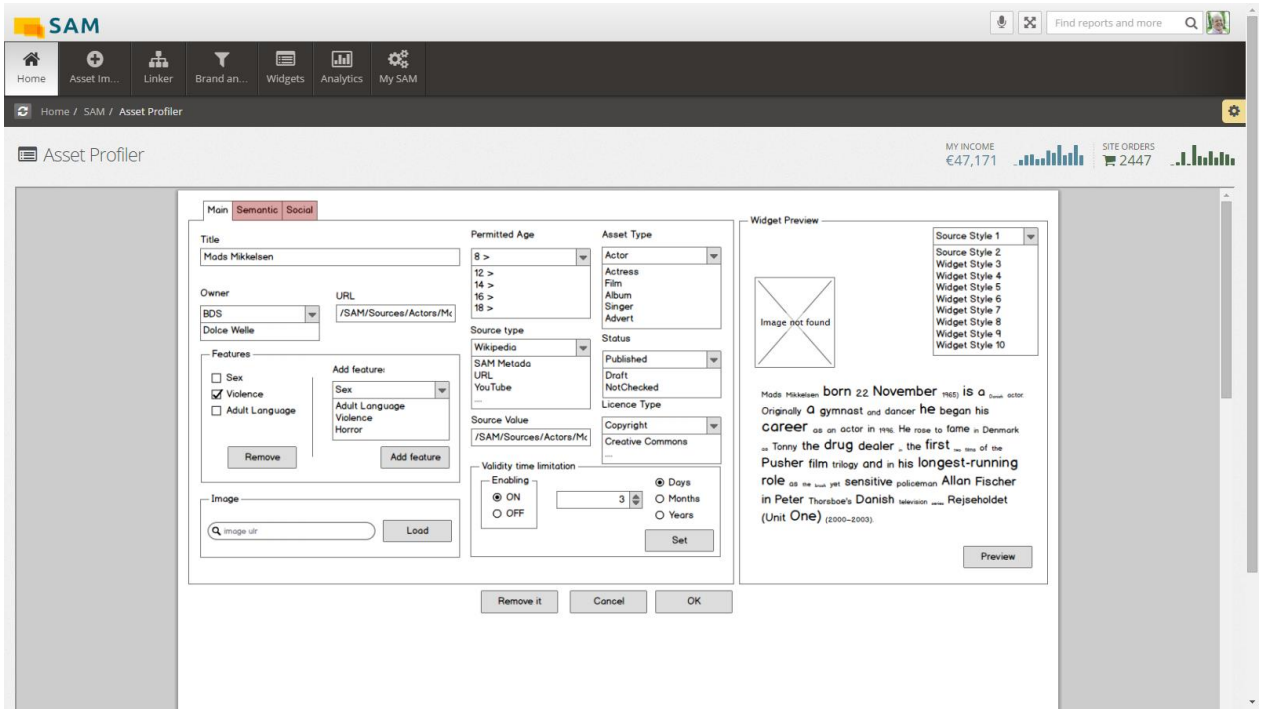

Figure 82: Marketplace – View/Edit Asset

6.1.4.1.6 Link Asset

This interface provides an area for aggregating and composing SAM Assets based on the preferences of the End User. In that sense, the End User is able to enrich his Assets with additional information coming from other assets, web or social network sources.

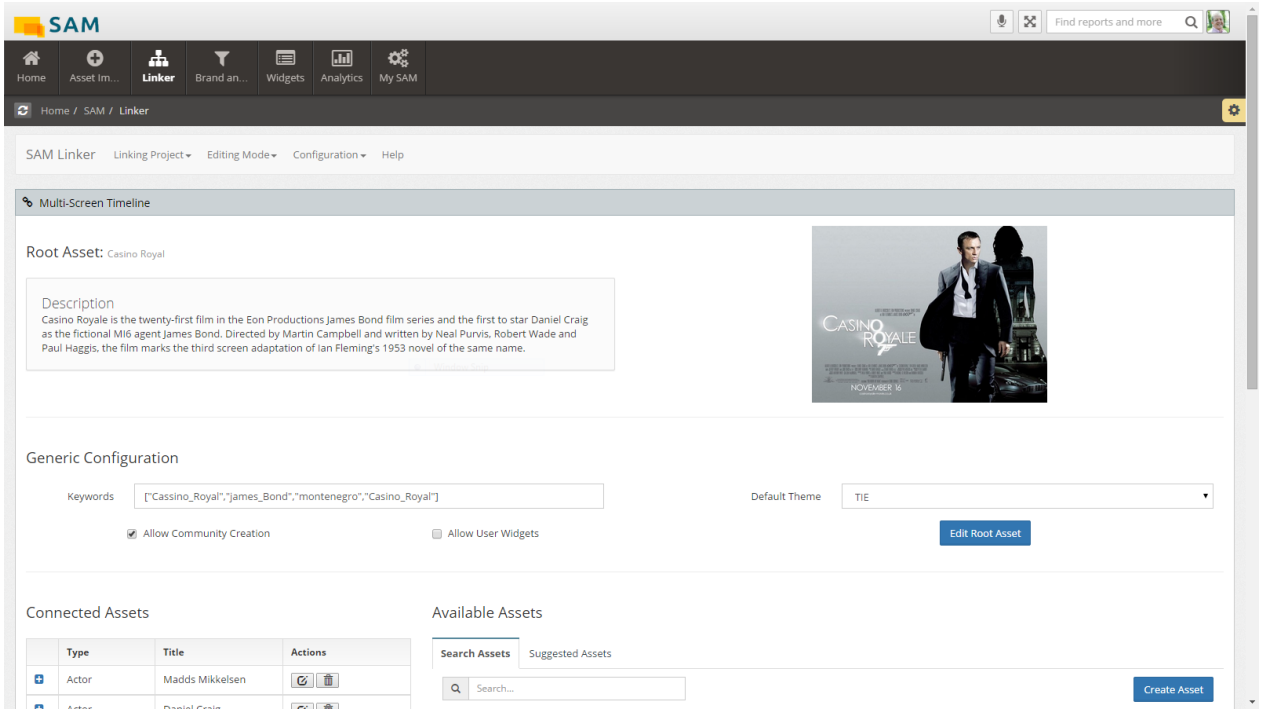

Figure 83: Marketplace – View/Edit Asset

6.1.4.1.7 Asset Import

This interface provides an area for End Users to interact in order to list/create/edit/schedule the different imports.

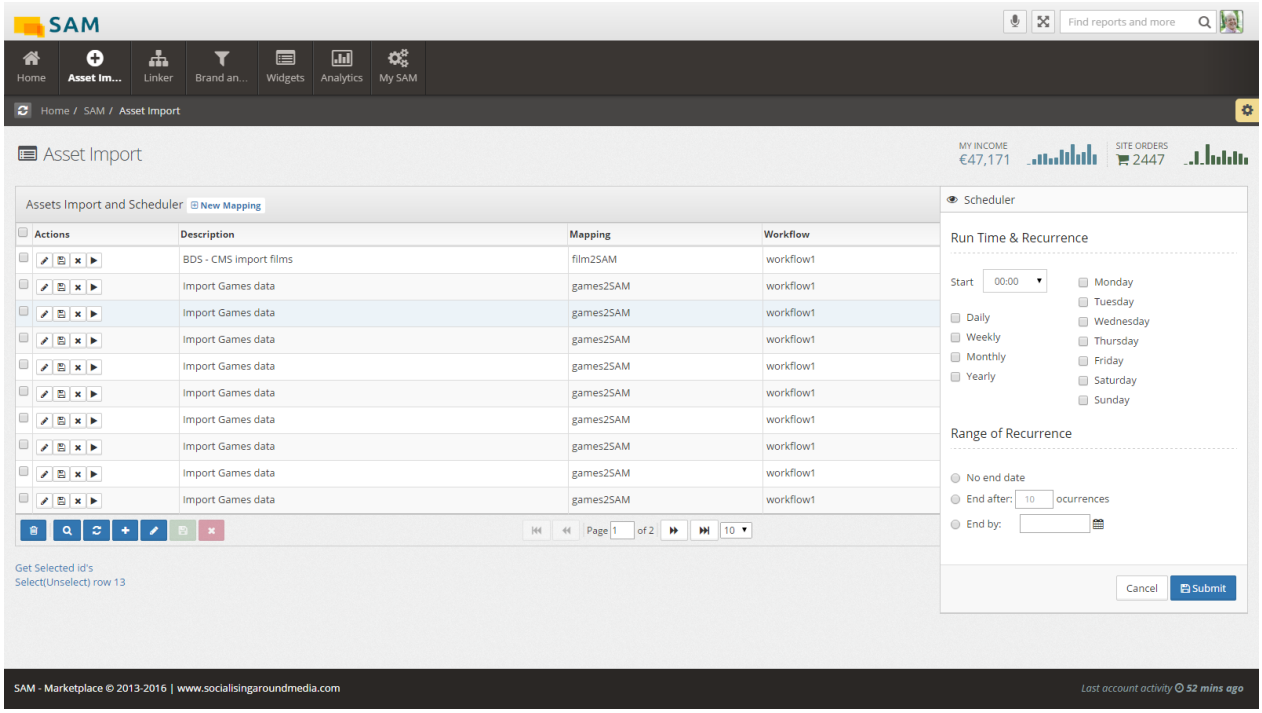

Figure 84: Marketplace – Asset Import

6.1.4.1.8 Brand and Consumer Protection

This interface provides an area for End Users to interact in order to list/create/edit the different Brand and Consumer Protection rules.

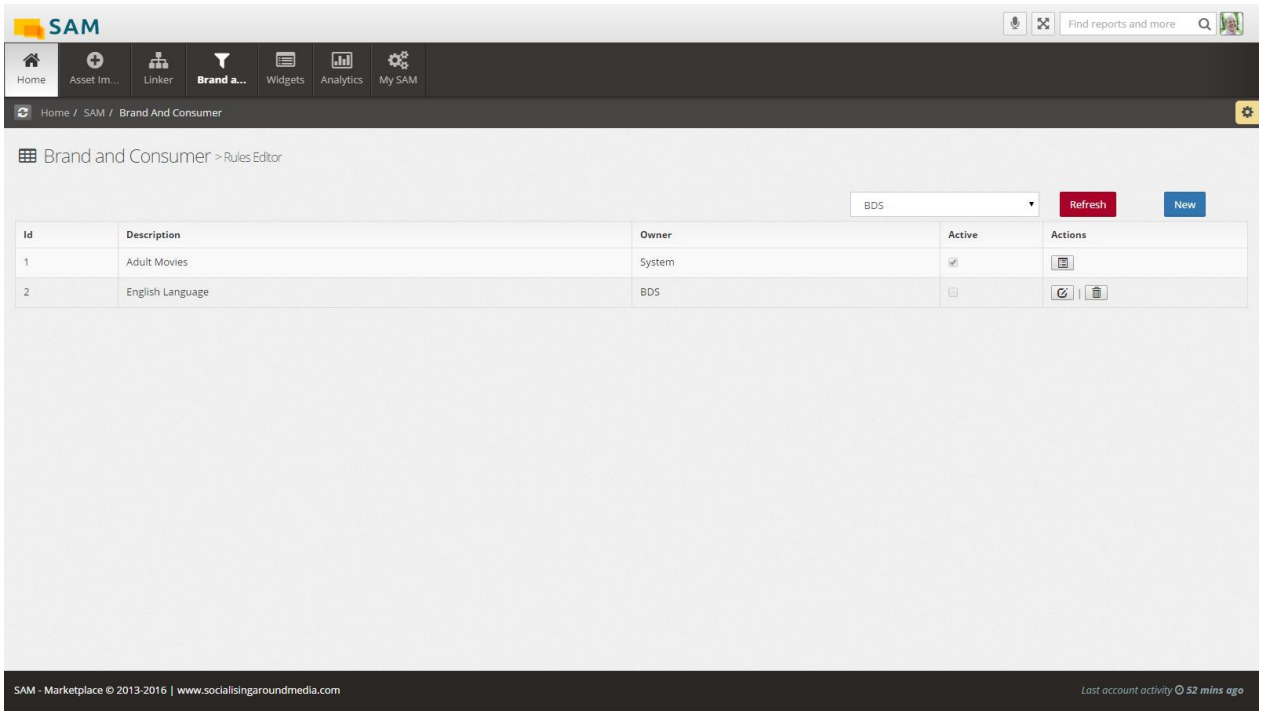

Figure 85: Marketplace – Brand and Consumer Protection

6.1.4.1.9 Widgets

This interface allows the End Users to create, edit or delete presentation templates for every asset the End User owns.

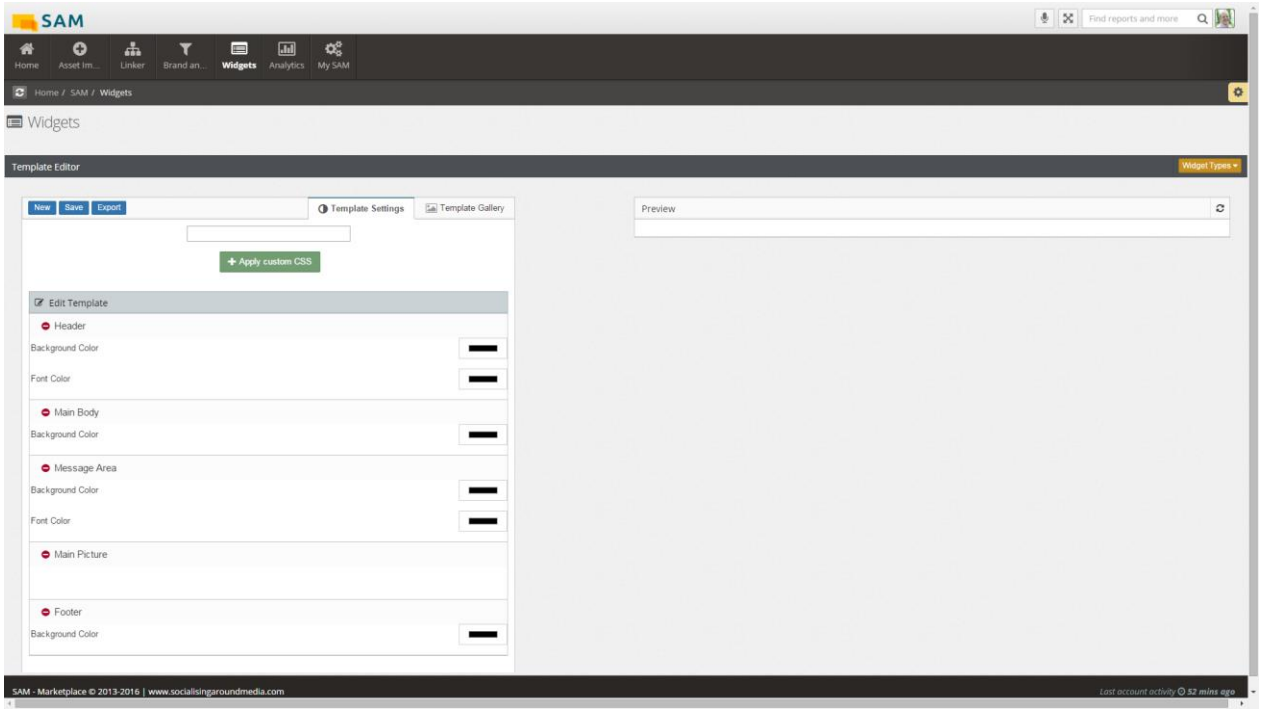

Figure 86: Marketplace – Brand and Consumer Protection

6.1.4.1.10 Analytics

This interface provides an area for providing business-related reports to the different stakeholders.

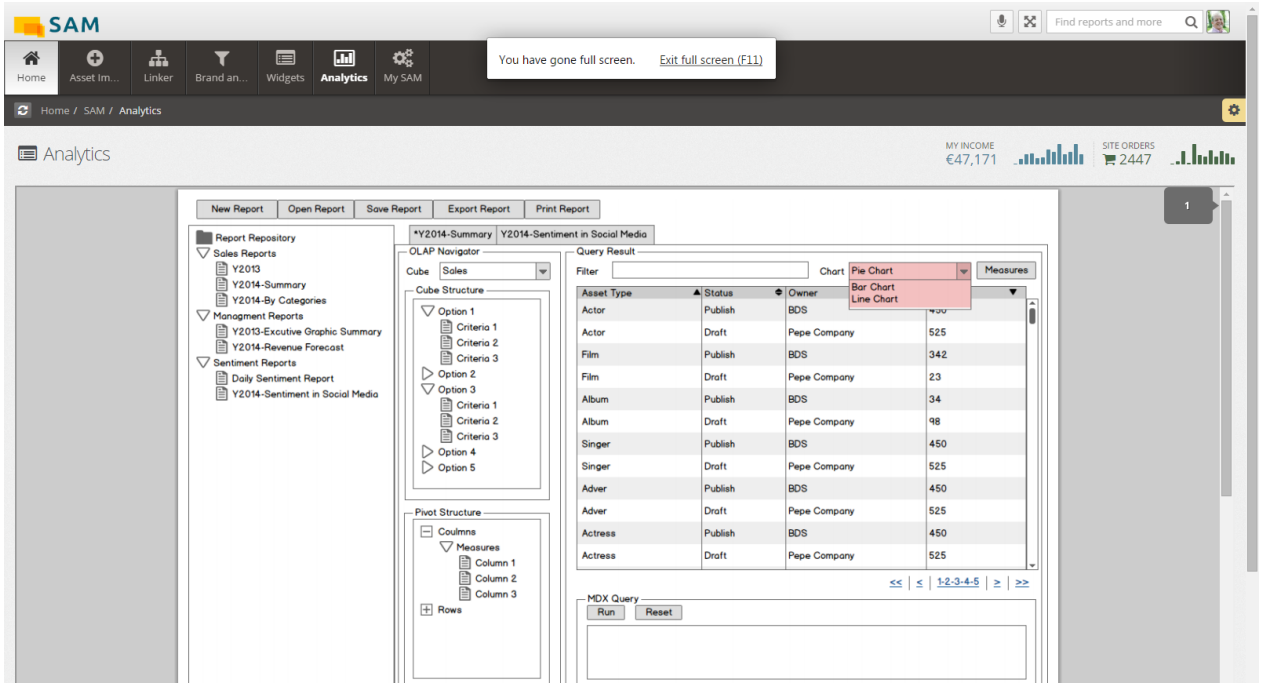

Figure 87: Marketplace – Analytics

6.1.4.1.11 My SAM

This area offers the functionalities to configure the personal information of each End User and contains the following subsections:

- Company Details
- Invoice History
- Payment Gateways
- Order Information
- Sales Information

6.1.4.1.11.1 Company Details

The User Account contains all personal information for End Users. The End User is able to update the information such as changing the password.

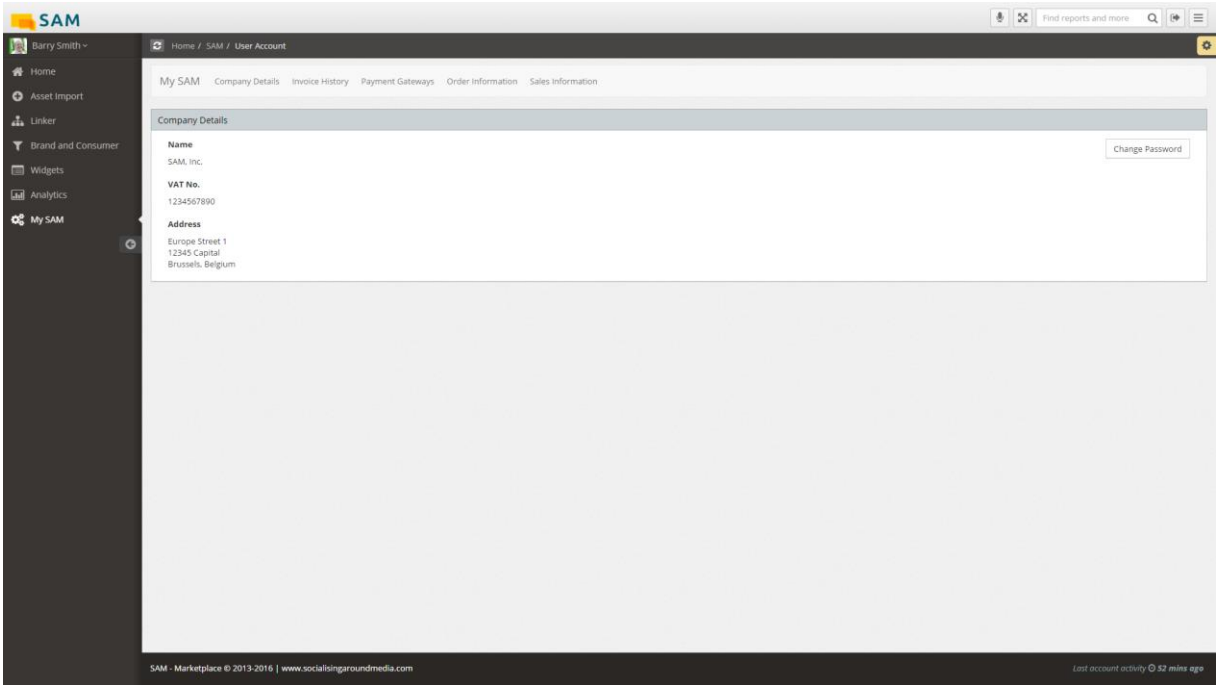

Figure 88: Marketplace – Company Details

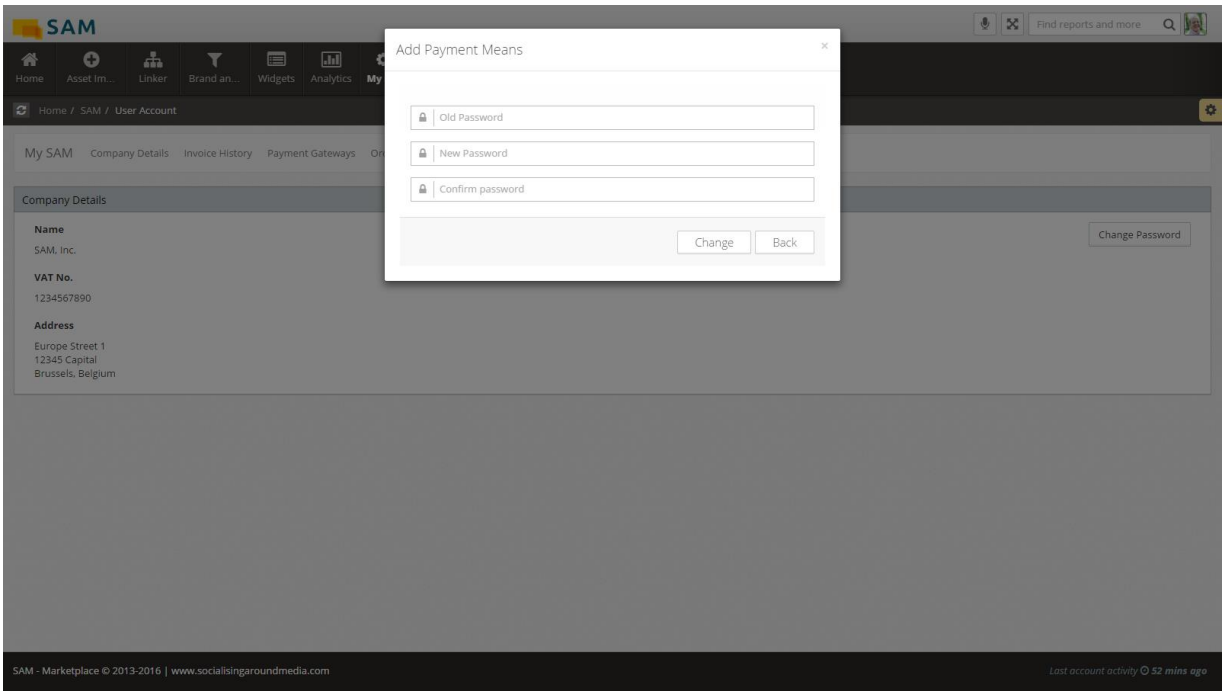

Figure 89: Marketplace – Change Password

6.1.4.1.11.2 Invoice History

This user interface permits functions such as list/filter/export for the different End User invoices

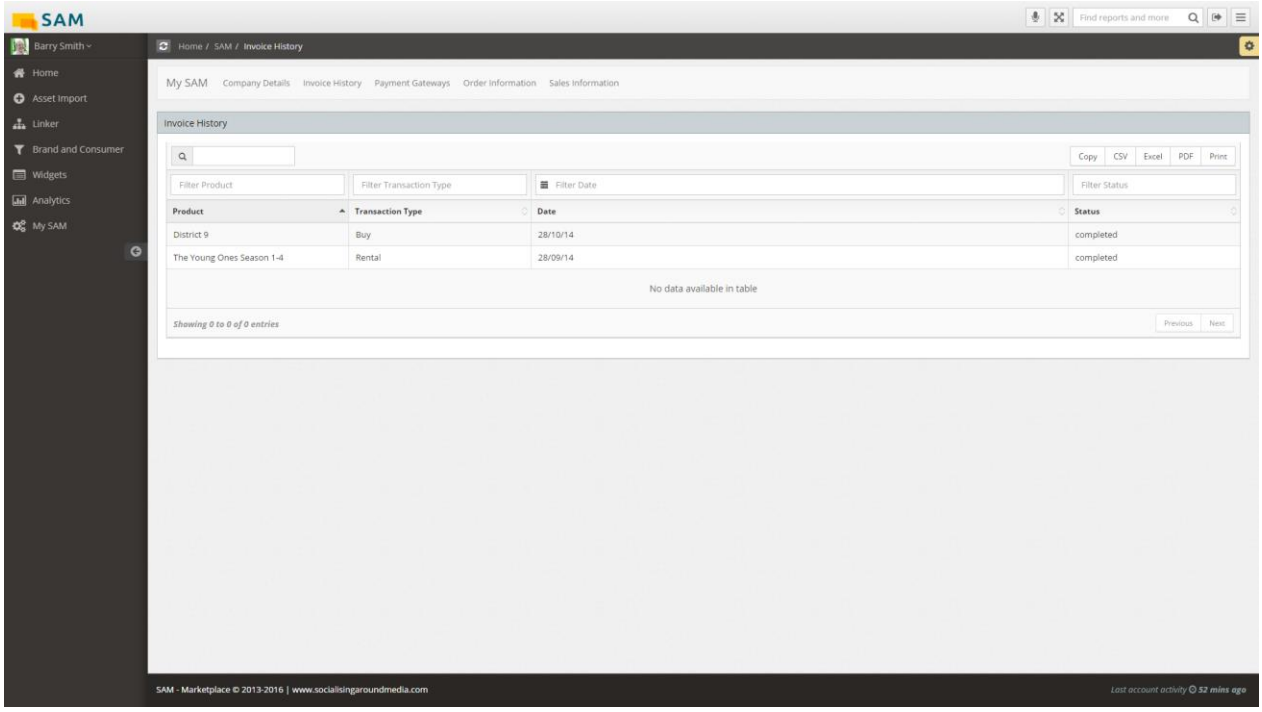

Figure 90: Marketplace – Invoice History

6.1.4.1.11.3 Payment Gateways

This interface allows list the different user payment methods (Figure 91) and functions such as add a new payment (Figure 92), edit payment (Figure 93) or delete payment for the different payments methods.

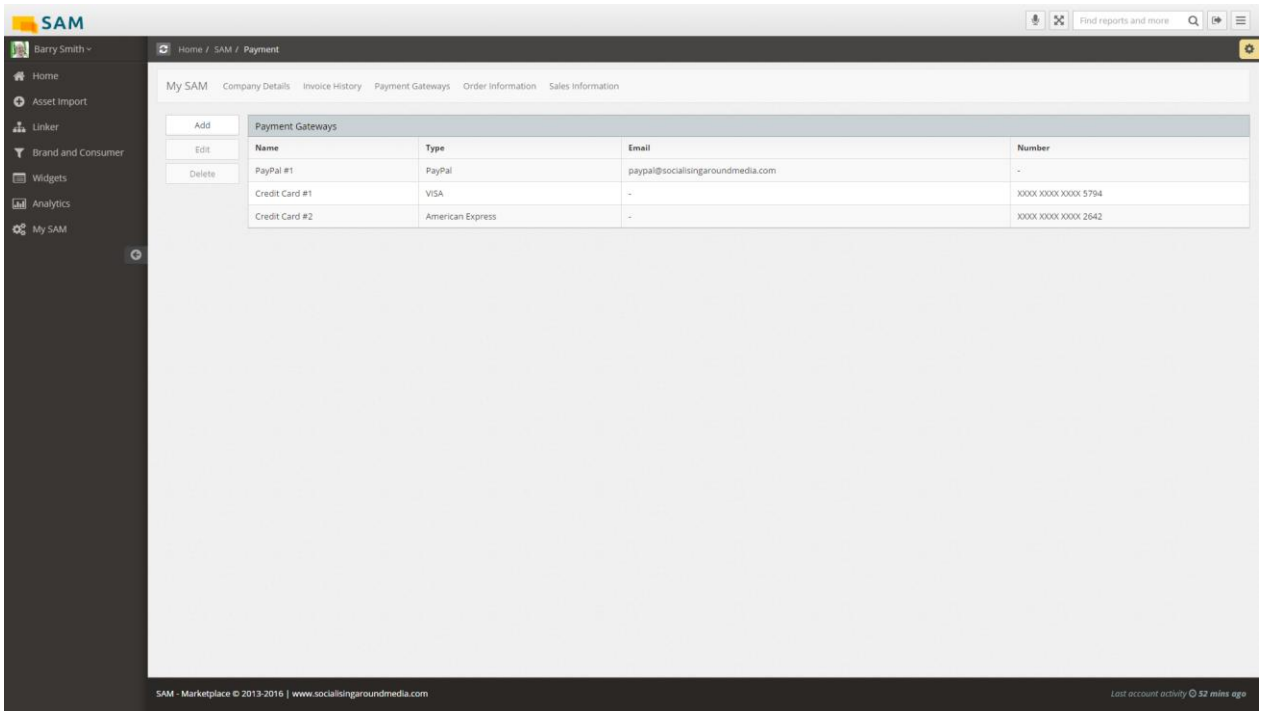

Figure 91: Marketplace – Payment Gateways

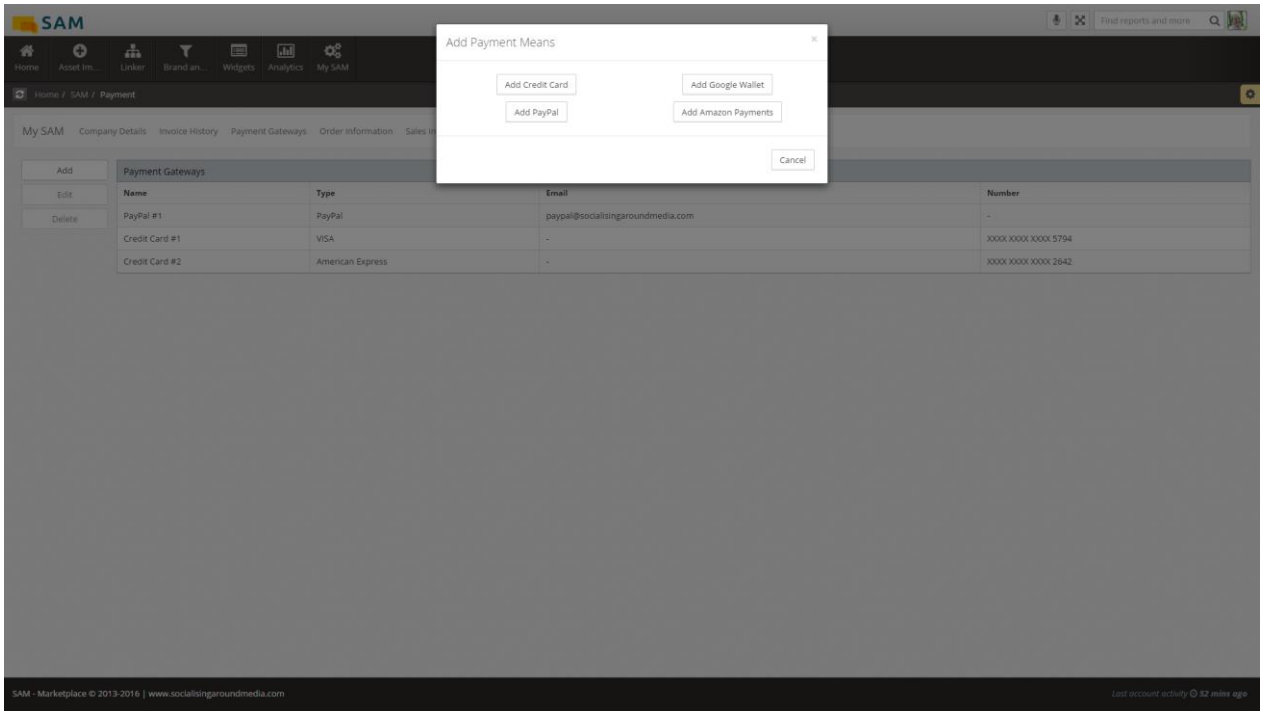

Figure 92: Marketplace – Add Payment Gateways

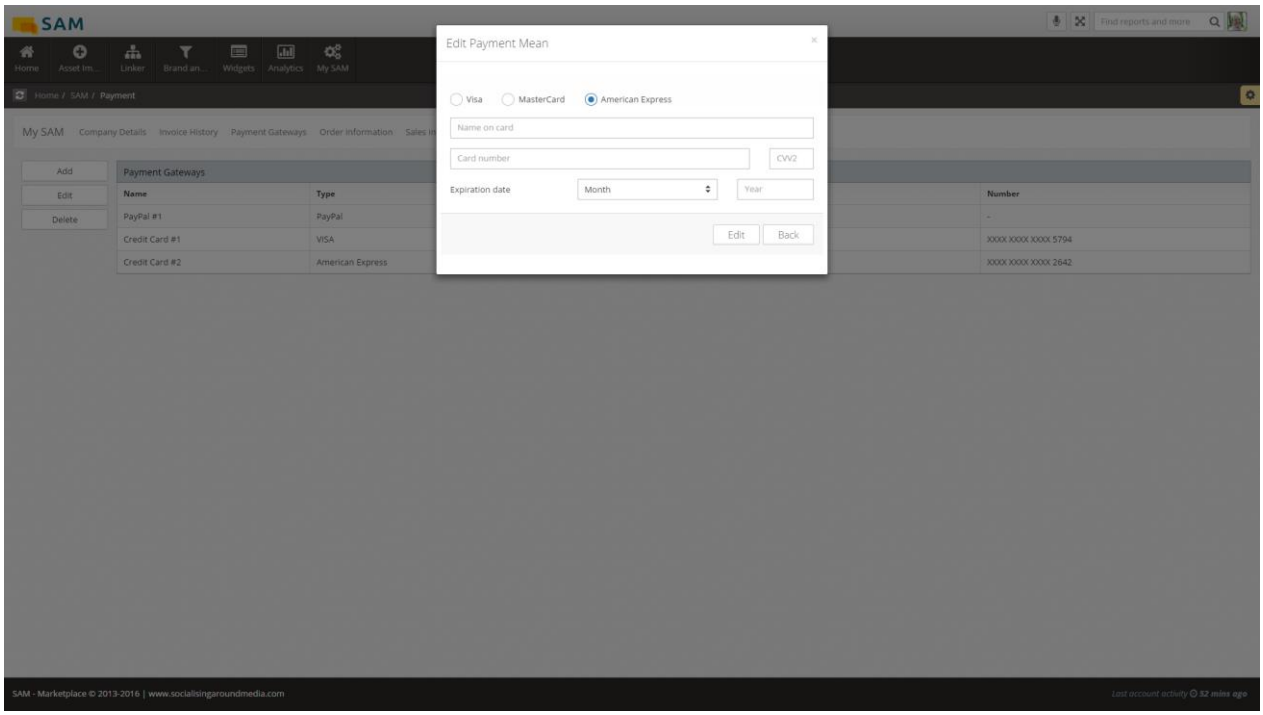

Figure 93: Marketplace – Edit Payment Gateways

6.1.4.1.11.4 Order Information

End Users can view all the orders, which were carried out. The End User is able to sort the listing and also view the status of every order. This guarantees a transparency about the orders. It is possible to see both the received and placed orders.

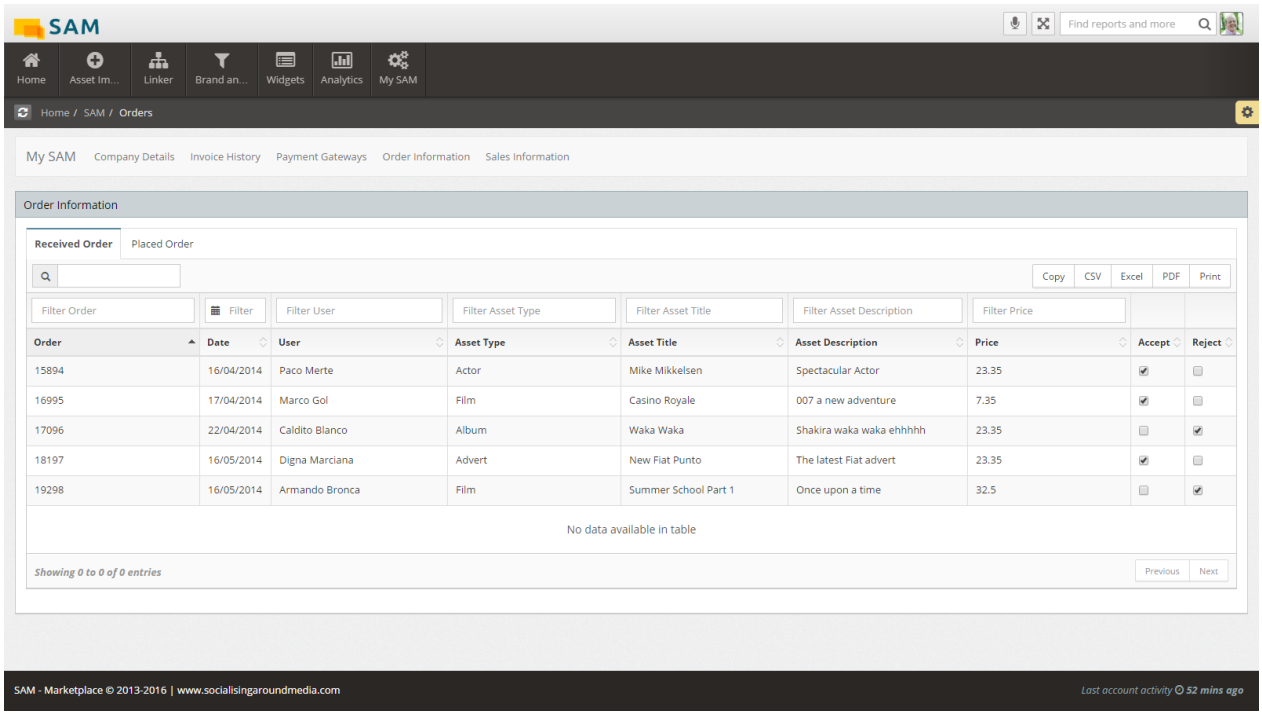

Figure 94: Marketplace – Received Orders

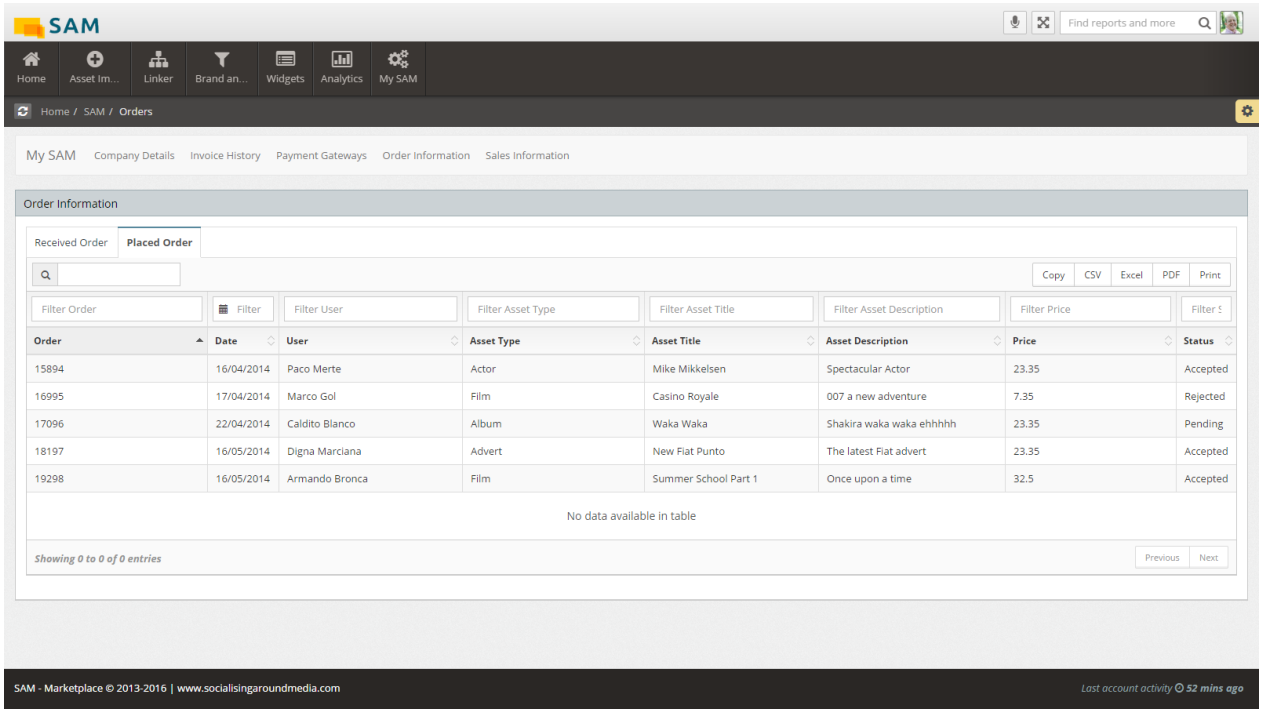

Figure 95: Marketplace – Placed Order

6.1.4.1.11.5 Sales Information

This interface provides an area to define the business models for any asset or service. Its various screens are:

- Sales Information. In this widget, both a list of uploaded assets (Figure 96) and a list of acquired assets (Figure 97) are shown
- Asset Information
  - General. The general information about the selected asset is shown
  - Distribution. The asset business models information is shown. This tab contains 2 widgets. The first one shows the business models itself, and the second one the different payment methods accepted (Figure 98)
  - Usage. In this tab, a list with the usage history of the asset is shown (Figure 99)
  - Content. In this tab, the asset content is shown (Figure 100)

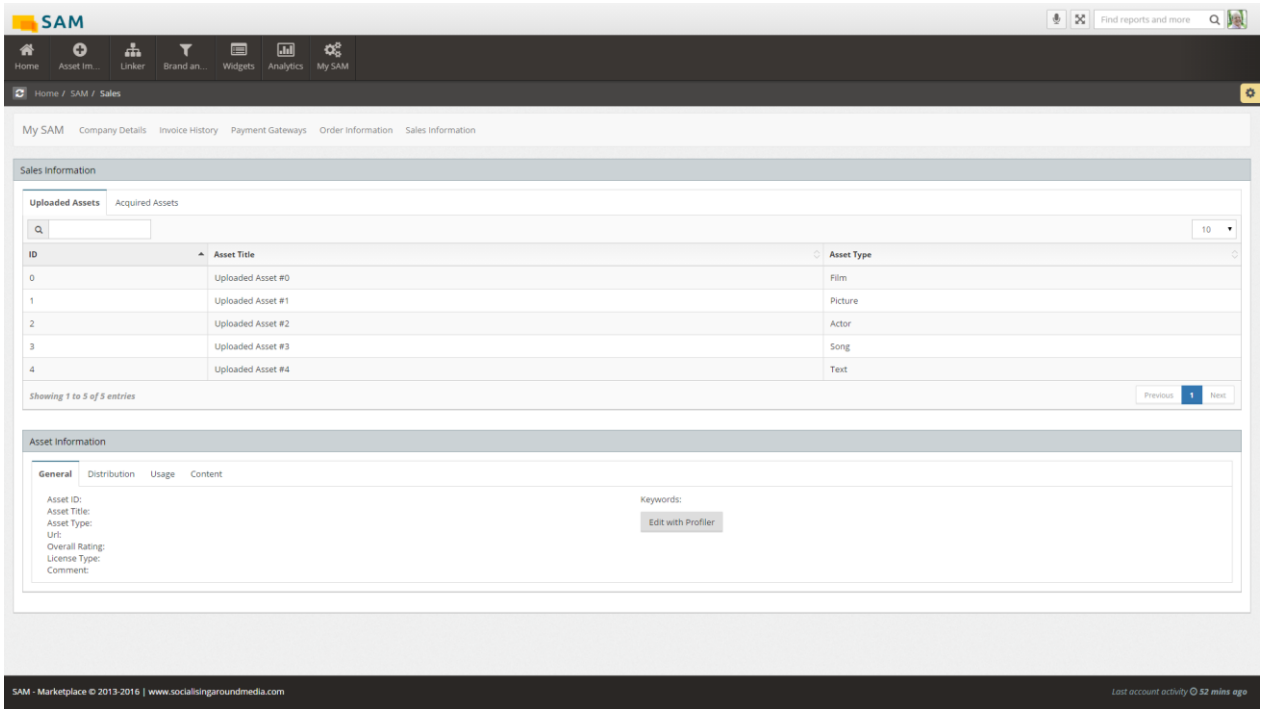

Figure 96: Marketplace – Sales Information Uploaded Assets

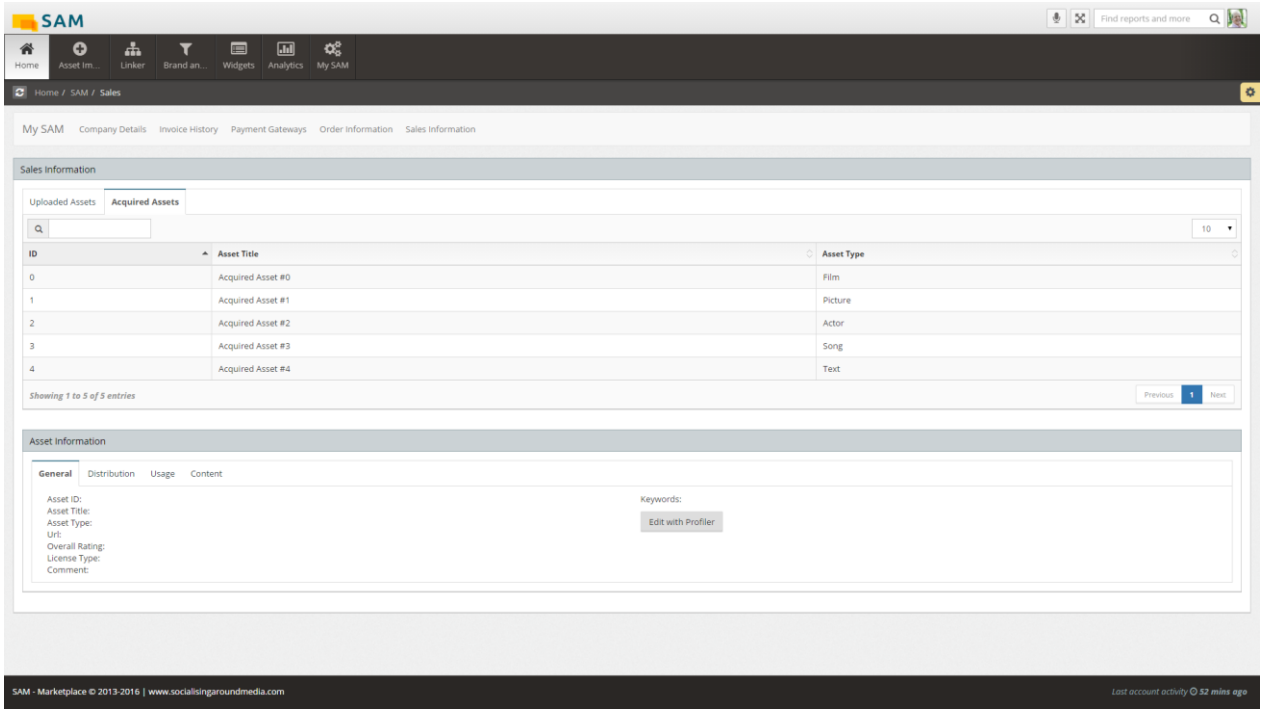

Figure 97: Marketplace – Sales Information Acquired Assets

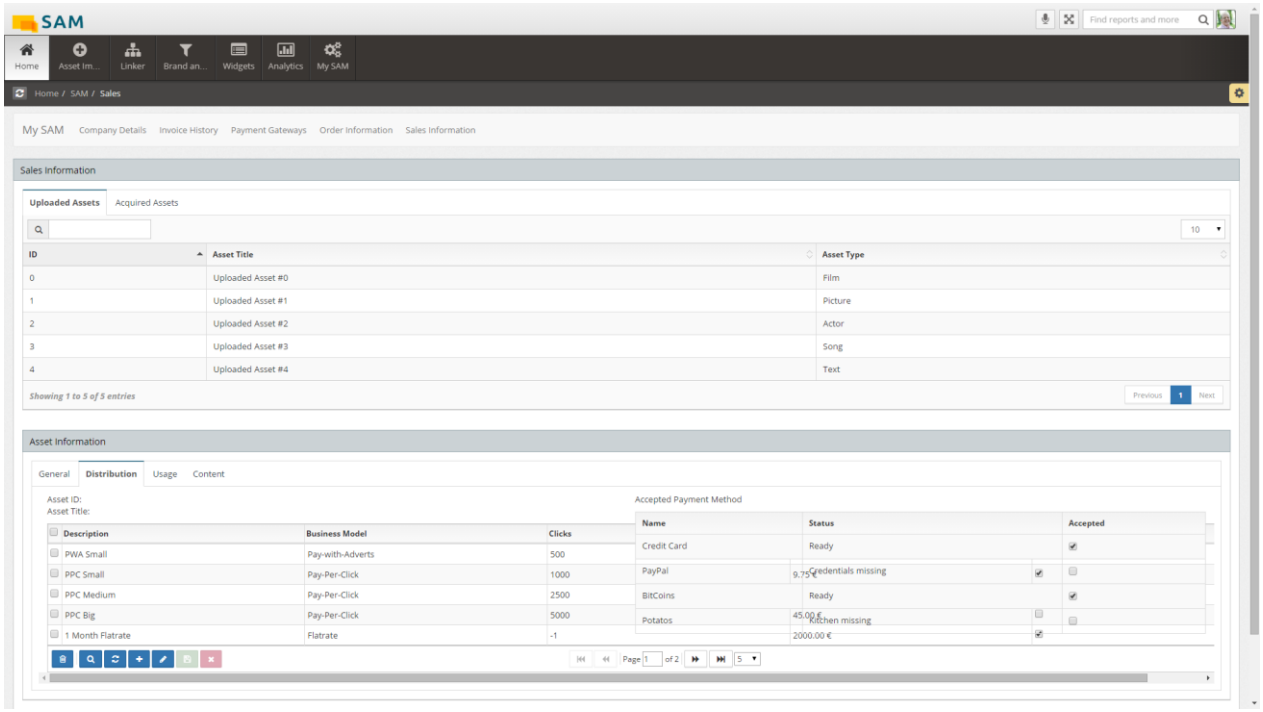

Figure 98: Marketplace – Sales Information Distribution

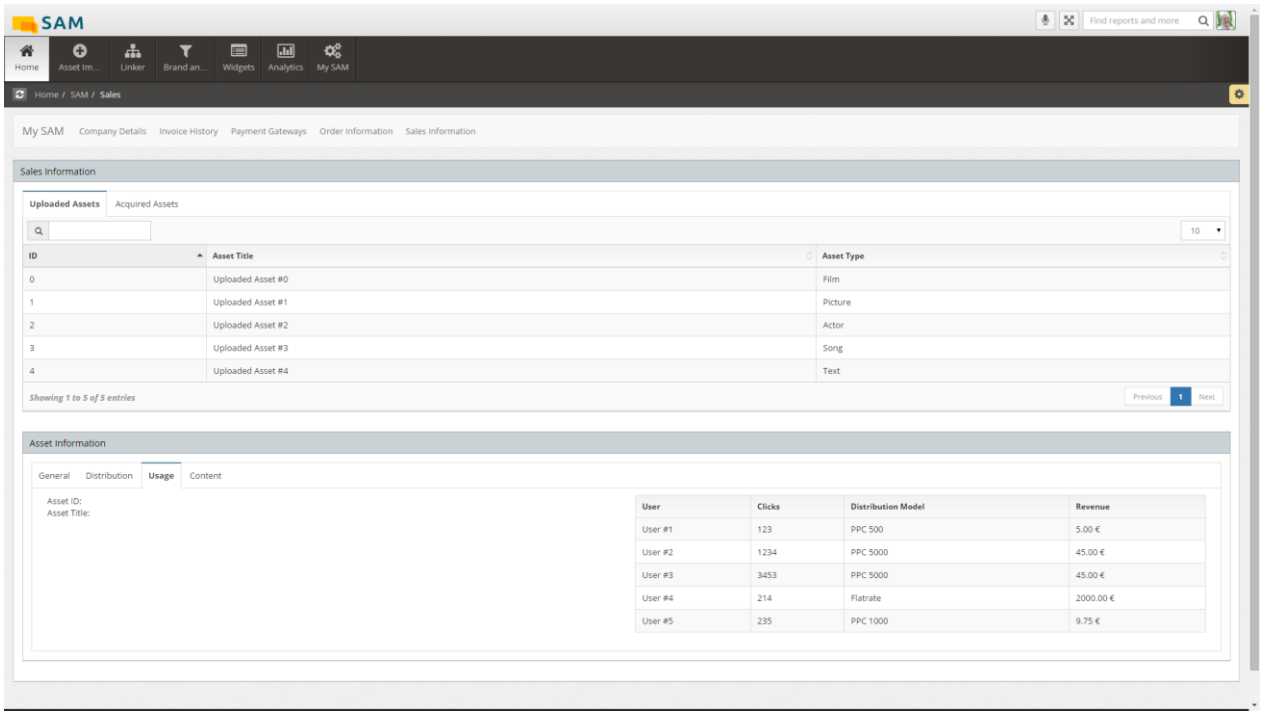

Figure 99: Marketplace – Sales Information Usage

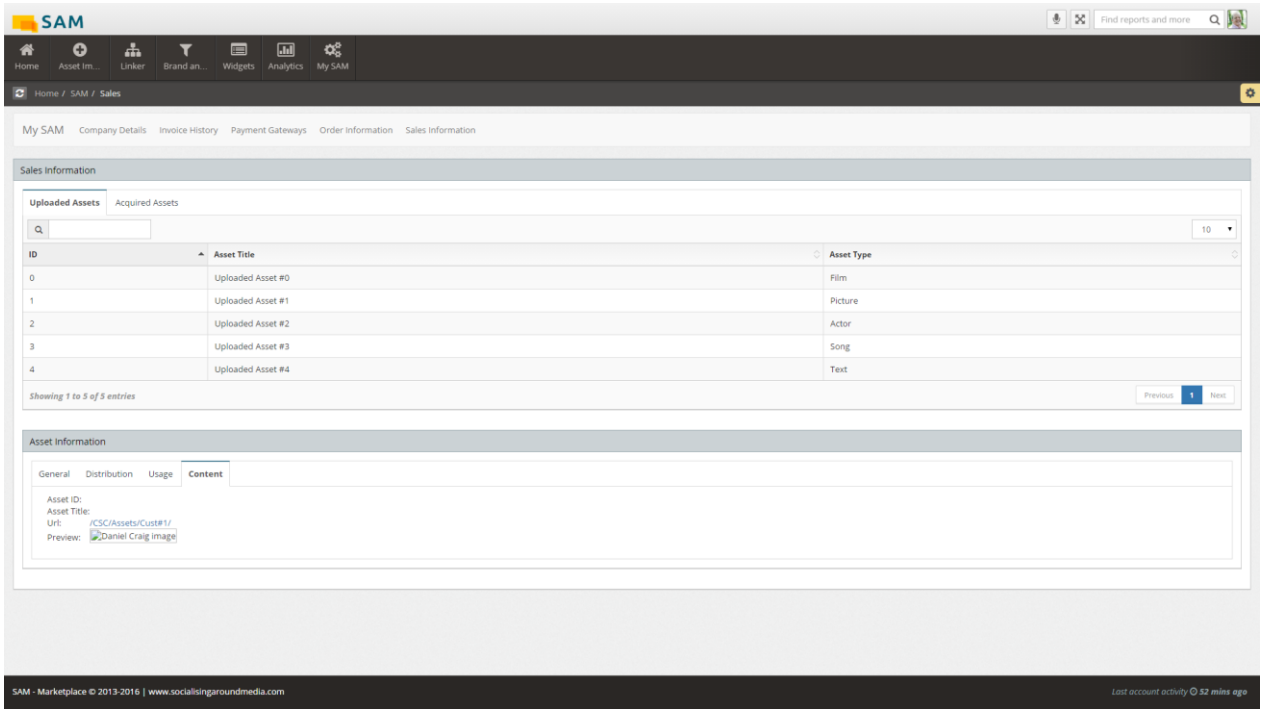

Figure 100: Marketplace – Sales Information Content

6.1.4.2 For Developers

6.1.5 Limitations and Further Developments

As this is the 1st Marketplace prototype there are still several limitations. The most important are the following:

- The My SAM and Dashboard functionalities are not connected with the marketplace web services
- The Analytics component and the Asset Profiler are not integrated yet.
- The connection with the Identity and Security Services component for data authorisation is not yet integrated, since this component is under development. The first prototype of T4.4 Distributed Identity, Trust and Security should be ready in M19 as specified in the DoW.

6.1.5.1 Prototype 2 Planned Tasks

Regarding the next steps in the Marketplace, Figure 101 and the following subsections summarise the tasks planned for the second prototype (to be delivered in M25).

Subcomponent	Task
Marketplace UI	• Polish the different user interfaces in order to provide a better user experience. • Implement the Identity and Security engine. • Develop the shopping cart web services to make progress in the development • Develop the necessary web services in order to implement all the functionalities of the My SAM area
Business Rules Manager	Develop the necessary web services to manage the business rules
Payment Gateways	Develop the engine to communicate with the different payment Services
Asset and Services Manager	Develop the engine to manage the requests from the End User through the Marketplace User Interface and the communication with the Federated Instances and Cloud Storage
Publisher	Develop the engine to publish assets and services
Syndicator Listener	Develop the engine to manage the requests from the Syndicator component

Figure 101: Tasks Planning for the Marketplace 2nd Prototype

6.1.6 Research Background

For the current prototype implementation and the overall approach the following papers have been taken into consideration:

Source	Subcomponent	Description
Design challenges for user-interface mashups: user control and usability in inter-widget communications	Marketplace UI	Based on the results of user studies on the OMELETTE mashup platform, this paper analyzes the problem space and evaluates possible solutions to improve user perception of IWC.

Figure 102: Research Background Marketplace

6.1.7 Target Performance

For this component the following key performance indicators (KPI) have been defined:

Topic	Description	Target KPI
Comprehension	The website should be legible	The language errors should be <1%
Navigability	Navigation is an essential element of the web, which can provide a great user experience, and even influence the KPIs. It is possible to influence and improve the interaction of End Users creating more revenue for SAM business	Tree testing[20]
Accessibility	The Marketplace should work in the most important browsers and it should be well formed XHTML markup and CSS markup	• The intention is to make available the Marketplace using Chrome and Opera • Pass the W3C validations
Website Speed	The time it takes for a website to load for visitors is very important. The visitors always expect a quick answer, for this reason the load page should be as fast as possible.	• The page load velocity should be <2 seconds • The average size of the download content should for page should be <100Kb
Searches time	The base of an online marketplace is the time that it takes for searches since it is the most common action in this kind of websites.	The time to give an answer for a search should be <2 sec

Figure 103: Target Performance Marketplace

6.1.8 Summary

This section provides a description of the first prototype of the Marketplace component developed in task T5.5 Marketplace. The main outcome of this task is the software of the Marketplace component. This prototype is the first of the three iterations planned for this component.

The most important goal reached during this 1st prototype has been the design and the development of a 1st version of the Marketplace User Interface. In the different sections it has been presented what the necessary requirements are for both users and developers in order to manage this user interface.

The last section was dedicated to describe the limitations of the current prototype, also describing the next steps considered for the second version of the component, which should be delivered in M25.

6.2 Syndicator – Data API Services

This section describes the SAM Data API Services first prototype functionalities.

6.2.1 Scope and Relationship

This subcomponent is part of the Syndicator component. It offers the means so that the different components and/or widgets can ask for information stored in SAM, taking into account the user context, the assets business constraints, etc. The Data API Service will syndicate the raw data and will interact with the Multi-device Representation component to provide the demanded information in the correct visual format. It will be also be accessed by 3rd Party systems Apps developed by external Software Providers (external systems). For accessing SAM information to be used i.e. by mobile apps, external web sites, etc.

[20] http://en.wikipedia.org/wiki/Tree_testing

This subcomponent offers the services API to distribute the syndicated information. The different components involved in this process can be seen in Figure 104.

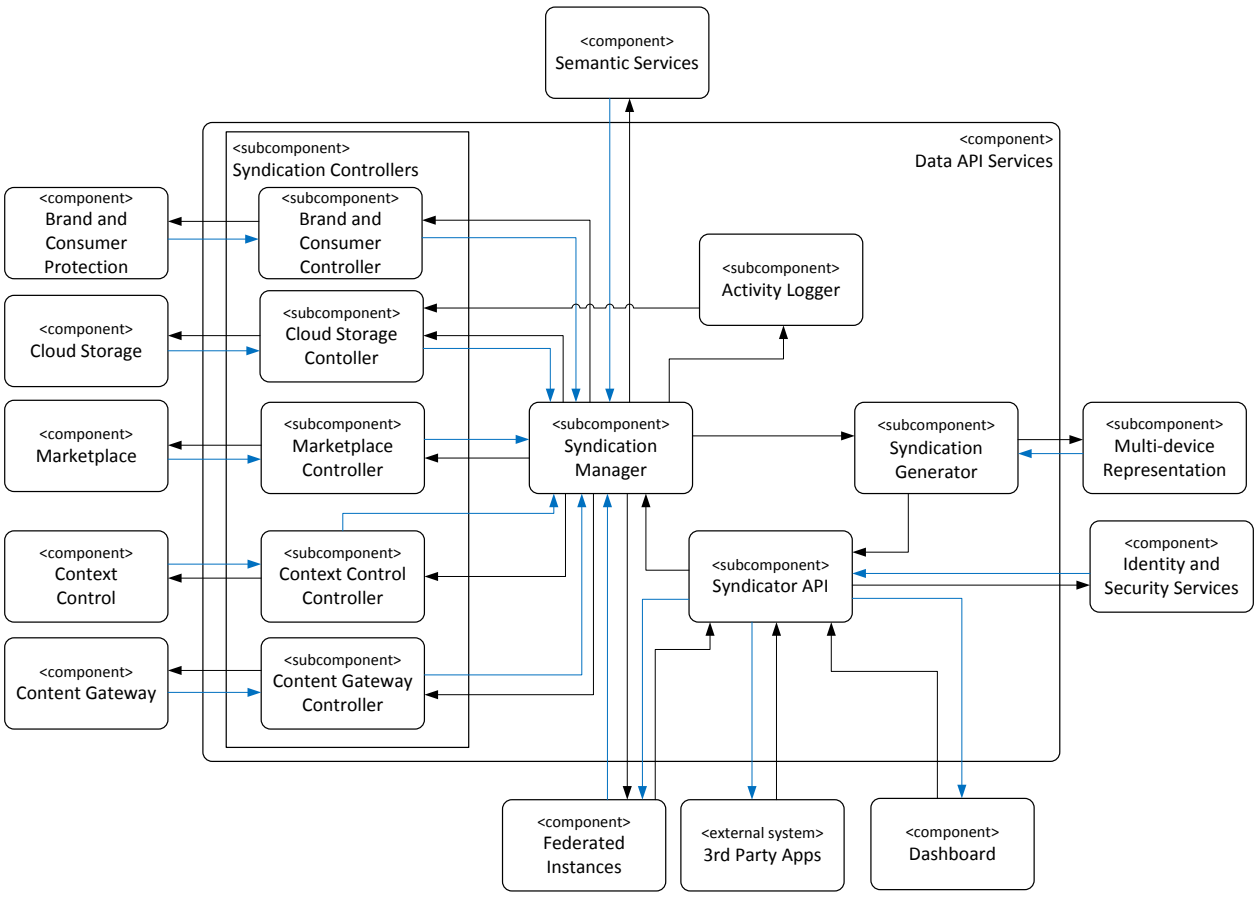

Figure 104: Data API Services

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.4.1 (Architecture), D3.2.2 Section 4.5.1 (Functional Specification) or D3.3.1 Section 3.5.1 (Technical Specification).

In order to implement this subcomponent, it was decided in the technical specification to extend and adapt the TIE Content Syndication Product (TIE CSP). Based on that decision, it has been carried out a research about the different TIE CSP services that can be adapted in order to offer the final widget syndication approach. Based on this approach, the strategy to extend or adapt the different modules has been designed for this 1st Prototype.

Based on this research, a 1st Syndicator API approach has been implemented. This subcomponent is in charge of the information syndication into the Dashboard or 3rd Party Apps. The 1st Prototype aims to show how this Syndicator API approach works based on a YouTube video about the film Casino Royal.

6.2.2 Requirements and Preparations

The Requirements and Preparations section describes the information needed to use the pilot, in terms of technical and non-technical requirements, software to be installed, basic knowledge, etc.

6.2.2.1 For Users

The user should be able to use some of the most important browsers. For this propose, the intention is to make available the Marketplace in Chrome and Opera. The development and primary testing has been done on Chrome.

6.2.2.2 For Developers

The Data API Services has been built with the .NET technology, thus Visual Studio IDE with the framework 4.5 and the IIS7.0 is required for the development.

6.2.3 Installation (Deployment)

The 1st prototype of the Data API Services has been deployed in an internal Server managed by TIEKinetix which fulfils the requirements detailed in section 6.2.2.2.

6.2.4 Execution and Usage

The Data API Services will be available via the web in the address specified in D5.5.1.

6.2.4.1 Client Side

To test the Syndicator API, the address specified in D5.5.1 should be used as part of Data API services and it should be made sure that these steps are followed.

1. Once the YouTube video has started a weather widget is shown.

Figure 105: SyndicatorAPI Demo – Weather Widget

2. After 35 seconds, The James Bond Twitter widget should be shown.

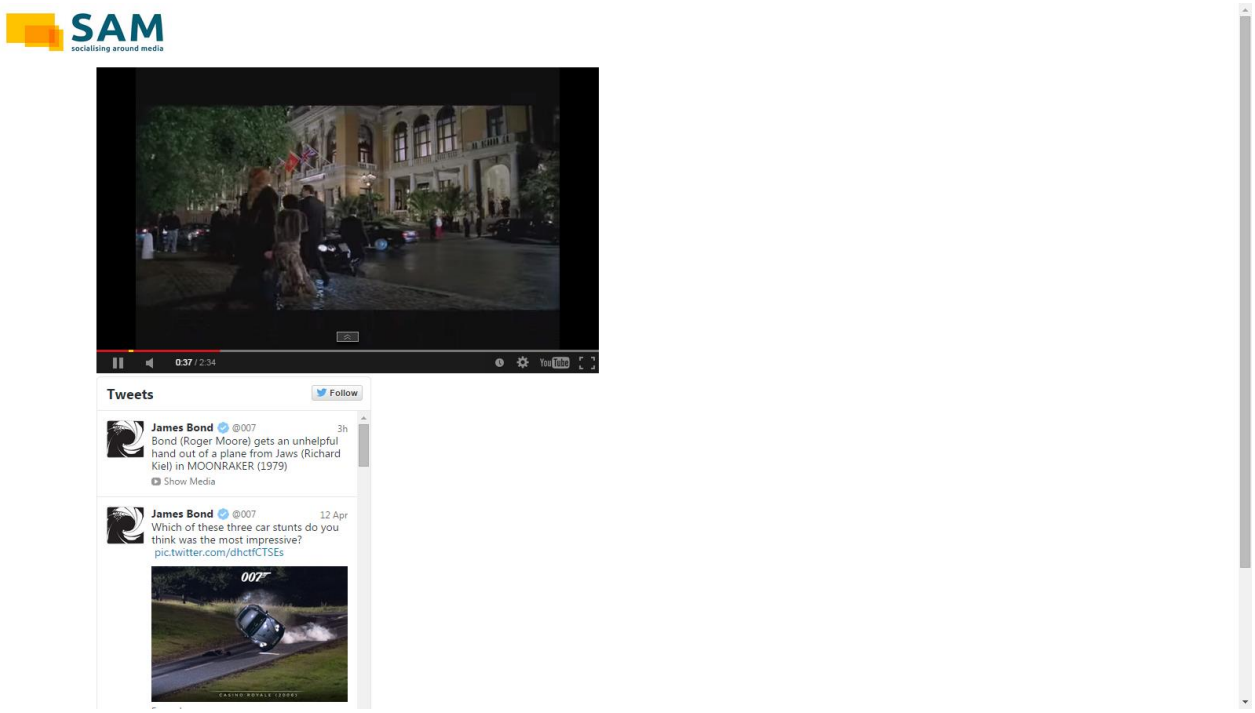

Figure 106: SyndicatorAPI Demo – Twitter Widget

6.2.5 Limitations and Further Developments

As this is the 1st Data API Services prototype there are still several limitations. The most important are the following:

- The SAM Server doesn't support .NET Framework
- The Syndicator API development is not connected with any SAM component and everything is based on sample information.
- The Data API Services is not connected with the Identity and Security Services, thus the data authorisation is not yet integrated since the latter is under development. The first prototype of T4.4 Distributed Identity, Trust and Security should be ready in M19 as specified in the DoW.

6.2.5.1 Prototype 2 Planned Tasks

Regarding the next steps in the Data API Services, Figure 107 and the following subsections summarise the tasks planned for the second prototype (to be delivered in M25).

Subcomponent	Task
Syndicator API	• Finalise the engine so that it communicates with the Dashboard and the 3rd Party Systems • Develop the integration with Identity and Security Services
Syndicator Manager	Develop the engine to coordinate the Syndication workflow.
Syndicator Generator	Develop the syndication engine and the communication with the Multi-Device Representation component.
Activity Logger	Develop the system to register all the activity that happens in the syndication workflow in order to check the user activity such as the number of clicks in an Asset per day.
Syndication Controller	Develop the different controllers to communicate the Data API Services with the necessary SAM component in order to filter the information to syndicate.

Figure 107: Tasks Planning for the Data API Services 2nd Prototype

6.2.6 Research Background

TIE Kinetix is a pioneer in the Content Syndication technology and a lot of articles and information about Content Syndication are published monthly by TIE Demand Generation department. Thus, this component has been designed based on a solid research and market background. It is also possible to find out more of this information in http://tiekinetix.com/en-us/resources.

6.2.7 Target Performance

For this component the following key performance indicators (KPI) have been defined:

Topic	Description	Target KPI
Syndication time	Time that is spent to syndicate the right information into the Dashboard and 3rd Party Systems.	The time to syndicate the widgets should be <2 sec.

Figure 108: Target Performance Data API Services

6.2.8 Summary

This section provides a description of the first prototype of the Data API Services component developed in task T5.5 Marketplace. The main outcome of this task is the software of the Data API Services component. This prototype is the first of the three iterations planned for this component.

The requirements have been presented necessary for both users and developers in order to manage the 1st version of the Syndicator API subcomponent in charge of the communication with Dashboard and 3rd Party Systems.

Also, the integration has started with the different TIE CSP modules CSP engine, Integration Engine and Asset Syndication.

The last section was dedicated to describing the limitations of the current prototype as well as the next steps considered for the second version of the component, which should be delivered in M25.

7 Document Summary

The current document presents the second version of this serie of deliverables (D5.9.x). An additional initial release (extra version) of the deliverable series D5.9.x was delivered by M15 (D5.9.0) in order to provide early information about the software deliverable D5.4.1 that was due in M15.

This document explains the different developments carried out for the 1st Prototype of the Content Gateways, Linker, Marketplace and Data API Services and for the 2nd Prototype of the Brand and Consumer Protection component.

Therefore this is the 1st Prototype for most of the components, for which several limitations were described. The following table shows the most important achievements:

Component	Goals
Content Gateways	• 1st version of the Semantic Editor Tool and progress in the mapping to transform BDS files into Syndicator format • 1st version of the Mapping Repository • 1st version of the Semantic Services component services necessary to carry out the Content Gateways functionalities. • 1st version of the Asset Import interfaces accessible through the Marketplace (See section 6.1.4.1.7)
Linker	• Create a module in the UI to support the timeline editing of the Assets • The basic module for Asset Linking, can now load connect and create Assets • The project manager is a UI module for accessing, saving and creating new Linking Projects • Define services for accessing, storing, deleting and creating new Assets
Brand and Consumer Protection	• Design and development of a new format for defining complex rules, specifying explicitly logical operators • Connection with the Cloud Storage to store and retrieve rules
Marketplace	• Design and develop a 1st version of the Marketplace User Interface • Design of the Marketplace Dashboard • Design of the Shopping Cart • Design of the Personal Area • Design of the Assets Business Rules area • Integration with Linker, Brand and Consumer Protection, Asset Import and Multi-Device Representation
Data API Services	• 1st version of the Syndicator API subcomponent in charge of the communication with Dashboard and 3rd Party Systems • Start of the integration of the TIE CSP (CSP engine, Integration Engine and Asset Syndication)

Figure 109: Task Summary D5.9.1

In conclusion, the different objectives established for this stage of the project have been reached. The work already carried out for all of the components provides solid pillars to reach the individual component objectives for the M25 developments in good pace.