



WP7 – Multi-Device Media Representation and Interaction

D7.9.3: Multi-Device Media Representation and Interaction Public Report (Third Version)

Deliverable Lead: TPVI

Contributing Partners: TPVI, ASC, TALK, TIE, NTUA

Delivery Date: 2016-10

Dissemination Level: Public

Final

This deliverable, the third of four deliverables, is the description of the third prototype implementations of the Tasks T7.1 SAM multi-device content and media representation, T7.2 SAM 2nd Screen media interaction, T7.3 SAM 1st Screen media interaction and T7.4 SAM multi-device dashboard. The deliverable covers the descriptions of the software deliverables in WP7. This document is a living document that is enhanced with each delivery of the different iterations of the WP7 prototypes.



Document Status	
Deliverable Lead	TP Vision
Internal Reviewer 1	BDS: Barry Smith
Internal Reviewer 2	UA: David Tomas
Type	Deliverable
Work Package	WP7 – Multi-Device Media Representation and Interaction
ID	D7.9.3: Multi-Device Media Representation and Interaction Public Report
Due Date	10.2016
Delivery Date	10.2016
Status	Final

Document History	
Versions	V1.0: including first and second reviews feedback

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 status and results of the different WP7 prototypes without exposing confidential information such as prototypes access information and source code. This confidential information can be found in the related Programme Participants Prototypes (D7.1.3, D7.2.3, D7.3.3 and D7.4.3).

This document is iteratively updated coinciding with the delivery of the different related software prototypes of WP7:

- SAM Multi-Device Content and Media Representation (T7.1)
- SAM 2nd Screen Media Interaction (T7.2)
- SAM 1st Screen Media Interaction (T7.3)
- SAM Multi-Device Dashboard (T7.4)

Each part is being developed iteratively with 3 prototypes, except for T7.4, which will have 4 prototypes, and will produce a software deliverable and an update of this public documentation (in the series D7.9.1 - .2 - .3 - .4).

As such, this third version of this deliverable (D7.9.3) describes the developments and results of the third iteration of the four prototypes in WP7. Although in the original DOW, T7.4 (SAM multi-device dashboard) was planned to start in M20, it was decided, and also indicated in D7.9.1, to start it in M14 because it has common parts to be used by both the 1st and 2nd Screen.

At this point in time the deliverable contains the following software prototype information:

Task	Component	Software Prototype Deliverable	Due	Section
T7.1	SAM Multi-Device Content and Media Representation	D7.1.3	M37	3
T7.3	SAM 1 st Screen Media Interaction	D7.3.3	M37	4

Figure 1: Overview of Tasks and Software Deliverables

This document is being complemented by D7.9.4, which contains the software prototype information for the components T7.2 – SAM 2nd Screen Media Interaction and T7.4 – SAM Multi-Device Dashboard, both of which have gone through further developments up to M37.

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

- **Scope and Relationship:** Describes the scope of the prototypes 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:** Depicts the current prototypes limitations
- **Research Background:** Presents relevant references to research publications
- **Target Performance:** Lists measurable targets for functions and usability
- **Summary:** Describes the conclusions of the implementation of the third prototype

Table of Contents

1	Introduction	7
1.1	SAM Project Overview	7
1.2	Deliverable Purpose, Scope and Context	7
1.3	Abbreviations and Glossary	8
1.4	Document Structure	8
1.5	External Annexes and Supporting Documents	9
2	WP7 Introduction	10
3	SAM Multi-Device Content and Media Representation	12
3.1	Scope and Relationship	12
3.1.1	First Prototype	13
3.1.2	Second Prototype	14
3.1.3	Third Prototype	14
3.2	Requirements and Preparations	15
3.2.1	For Users	15
3.2.2	For Developers	15
3.3	Installation (Deployment)	15
3.4	Execution and Usage	15
3.4.1	For Users	16
3.4.2	For Developers	21
3.5	Limitations	25
3.6	Research Background	26
3.7	Target Performance	26
3.7.1	Component KPIs	26
3.7.2	User Experience Measurements Tasks	28
3.8	Summary	28
4	SAM 1 st Screen Media Interaction	29
4.1	Scope and Relationship	29
4.1.1	1 st and 2 nd Screen Interaction	31
4.2	Requirements and Preparations	32
4.2.1	For Users	32
4.2.2	For Developers	33
4.3	Installation (Deployment)	33
4.3.1	For Users	33
4.3.2	For Developers	34
4.4	Execution and Usage	34
4.4.1	For Users	34
4.4.2	For Developers	36
4.5	Limitations	36
4.5.1	Possible further improvements	36
4.6	Research Background	36
4.7	Target Performance	37
4.7.1	Component KPIs	37
4.7.2	User Experience Measurements Tasks	37
4.8	Summary	38
5	Document Summary	39
	References	40
	Annex A: User Research	41
	Generic Target Performance KPIs	41

Testing Procedure41
Graphical Editor User Trials Scenarios.....42

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 WP7 tasks T7.1 SAM multi-device content and media representation, T7.2 SAM 2nd Screen media interaction, T7.3 SAM 1st Screen media interaction and T7.4 SAM multi-device dashboard. Each task will contribute different components to the SAM architecture that are developed

iteratively in 3 phases as per milestones 3/4/5 at M19/25/37 for tasks T7.1 and T7.3 and in 4 phases as per milestones 3/4/5/6 at M19/25/31/37 for T7.2 and T7.4 and will produce a software deliverable and an update of this public documentation (D7.9.x) with the timings indicated in the following table:

Deliverable	Date
D7.9.1	M19
D7.9.2	M25
D7.9.3	M37
D7.9.4	M37

Figure 2: Deliverable Schedule

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 user guide for the current status of the development.

This document focuses on T7.1 and T7.3. It is complemented by D7.9.4 that covers T7.2 and T7.4. Document Status and Target Audience

This document is the third iteration of the D7.9.x series and is listed in the DOW as public. It is primarily aimed at the project partners as a user guide but it also presents status information of the prototypes of the software components of WP7 to the interested public.

1.3 Abbreviations and Glossary

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

1.4 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 main objectives, constraints and status
- **Section 2 (WP7 Introduction):** Provides an overview of WP7 goals and the WP7 prototypes
- **Section 3 (SAM Multi-Device Content and Media Representation):** Describes the latest software deliverable developed in T7.1
- **Section 4 (SAM 1st Screen Media Interaction):** Describes the latest software deliverable developed in T7.3
- **Section 5 (Document Summary):** Briefly summarises the work presented at the deliverable, as well as the overall WP7 status

In Sections 3 and 4, for each component in the SAM Architecture, the following subsections are provided:

- **Scope and Relationship:** Describes the scope of the prototype implementation, its purpose and the main relationships with other modules 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 de different screens and actions implemented at the prototype itself, how to access it, and how to test the different implemented options.
- **Limitations:** Depicts the current prototype limitations and the expected improvements.
- **Research Background:** Presents relevant references to research papers and publications
- **Target Performance:** Lists measurable targets for functions and usability
- **Summary:** Describes the conclusions of the implementation of the third prototype.

1.5 External Annexes and Supporting Documents

- D7.1.3: Multi-Device Content & Media Representation
- D7.2.3: 2nd Screen Media Interaction
- D7.3.3: 1st Screen Media Interaction
- D7.4.3: Multi-Device Dashboard
- D7.9.4: Multi-Device Media Representation and Interaction Public Report (Final Version)

2 WP7 Introduction

WP7 is concerned with the multi-device representation and interaction with Asset content taking into account the wide spectrum of devices with different specifications in the market.

Specific objectives of this WP include:

- To implement a framework for multi-device based media representation (T7.1)
- To provide implementations of advanced user interaction using voice recognition, Inter-Widget-Communication (IWC) and 1st Screen component detection (T7.2)
- To produce a multi-device dashboard supporting media representation and advanced user interaction (T7.3, T7.4)

The results of WP7 are a set of related applications where the consumer/user can experience all the other SAM RTD WPs. Each component is developed iteratively as per milestones 3/4/5 (for T7.1 and 7.3) at M19/25/37 and 3/4/5/6 (for 7.2. and 7.4) at M19/25/37/37 and will produce a software deliverable and an update of the public documentation (D7.9.x – this document series).

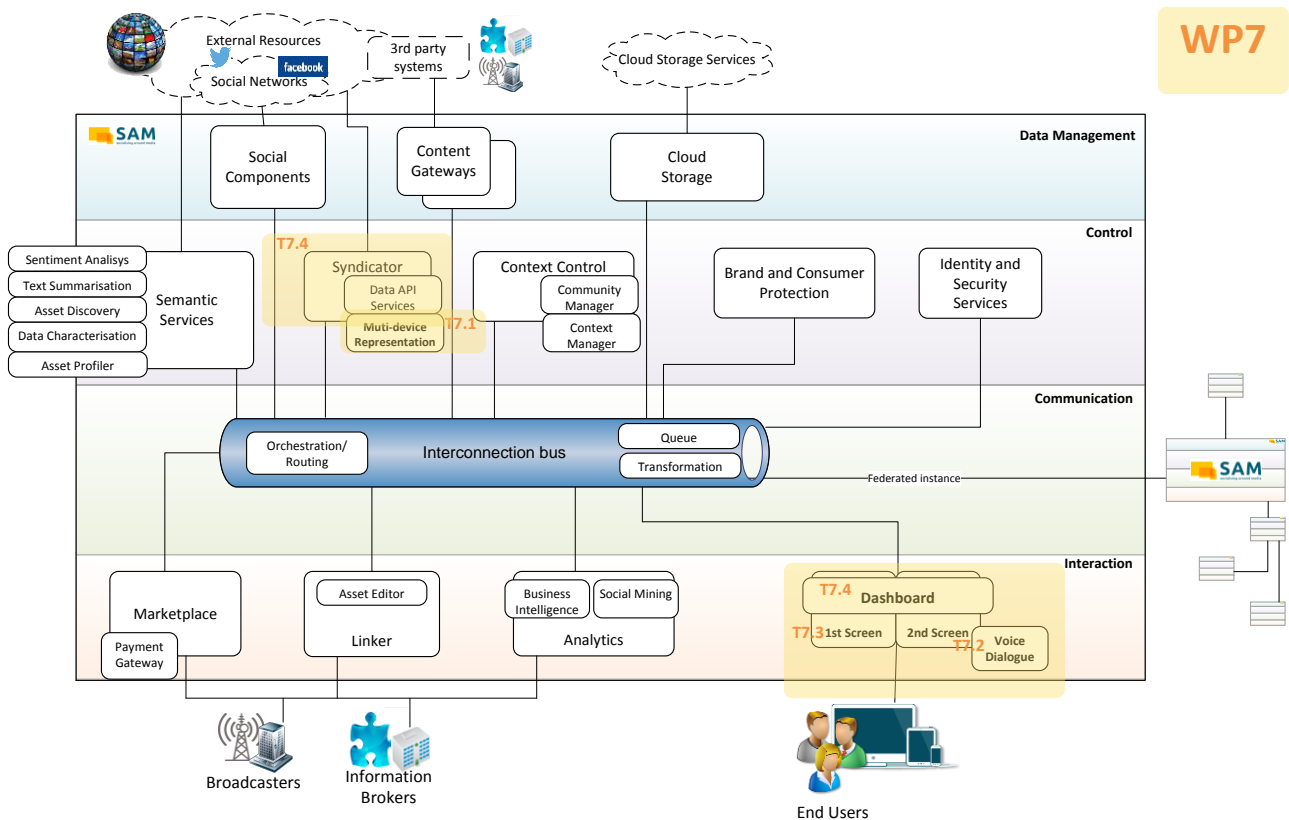


Figure 3: WP7 Components Contribution to SAM Architecture

The work in WP7 (as well as in the other development WPs) is managed by using the agile SCRUM methodology. For that purpose, a dedicated WP7 SCRUM board has been created in the SAM Jira task management system, with a representative of the WP Lead (TPVI) as the Scrum Master.

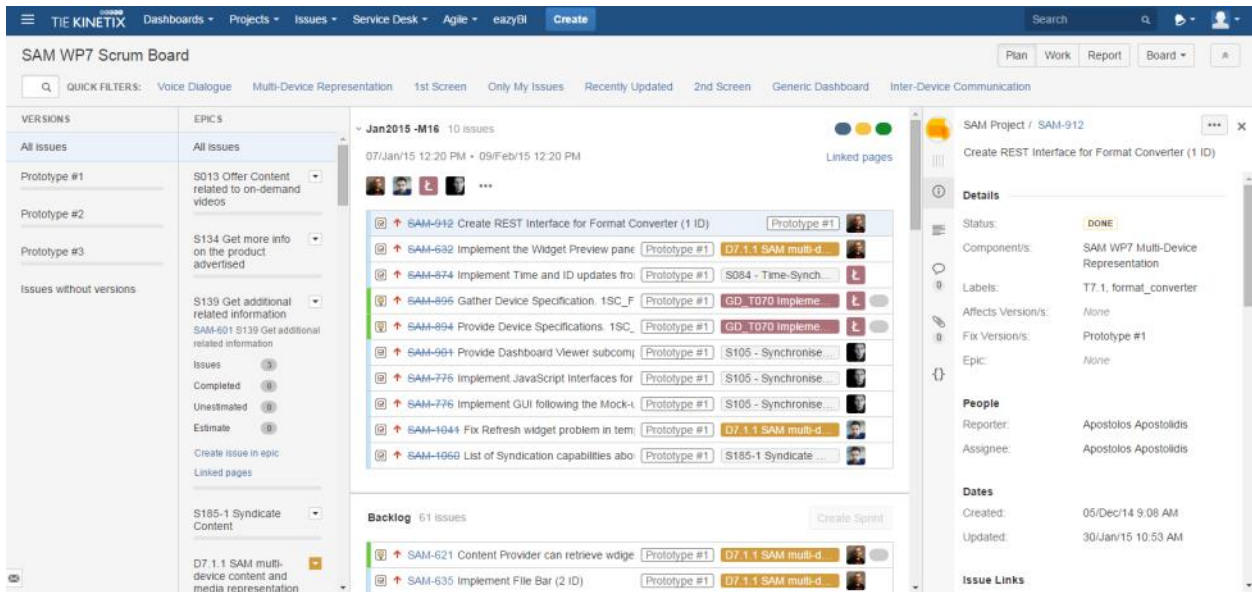


Figure 4: WP7 Scrum Board

Sprints are planned and executed monthly, and 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 unfinished work from the previous one. A Retrospective or Review 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.

During the development for the third Prototype of the different components developed in this WP, two technical meetings were held with all technology partners in which much progress was made and many of the integration and stability issues were tackled.

3 SAM Multi-Device Content and Media Representation

This section describes the software deliverable D7.1.3, which is the third prototype release of the SAM Multi-Device Content and Media Representation component.

3.1 Scope and Relationship

The Multi-Device Media Representation (MDR) component consists of two subcomponents: the Graphical Editor and the Format Converter. The former is active during Production Time while the latter during Prosumption.

Production Time is defined as the phase where Content Providers and SAM Administrators curate and manage content that will be available to the End Users. This content is syndicated to all End Users presented in the form of widgets. Each piece of information will be wrapped in an appropriate widget type. The Prosumption is the phase where the End Users consume and interact with the data that are supplied to them.

The Graphical Editor aims to enable both the SAM Administrator and Content Owner to modify the style and functionality of the widgets that will be used to syndicate information during the Prosumption.

The Format Converter is queried by any SAM component, which needs information about widget types and their content.

Figure 5 shows the two subcomponents of MDR and the logical connections that have been established between them.

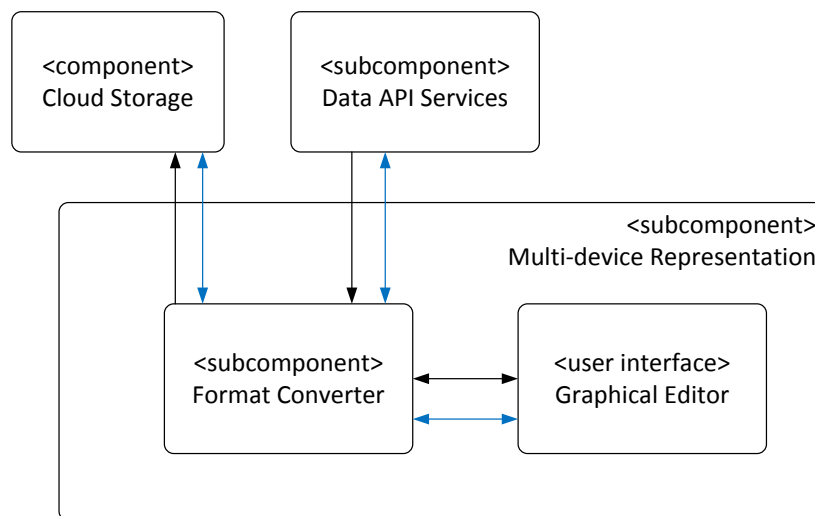


Figure 5: Multi-Device Representation (MDR) Architecture

For further descriptions of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.4.2 (Architecture), D3.2.2 Section 4.5.2 (Functional Specification) and D3.3.1 Section 3.5.2 (Technical Specification).

The third Format Converter prototype presents some alterations compared to the previous one. It no longer injects Asset Data into widgets but rather provides the Syndicator with CSS and HTML files for it to combine them with the Asset data. Specifically, the 3rd prototype provides the following improvements:

- Provide the Syndicator with the currently active CSS style for a specific widget type
- Provide Syndicator the HTML style for a specific widget type

- Provide Widget Editor and Linker with a list of all widget types
- Provide Widget Editor with the ID of the default CSS style for a specific widget type
- Provide Syndicator and the Linker with Asset field – Widget part relations

As can be seen in the list above, the Syndicator can retrieve the CSS and HTML for any given widget type from the Format Converter in order to be able to send it for displaying on the 1st and 2nd Screen. Likewise, the Linker and the Widget Editor need to retrieve a list with all the active widget types that a user can configure. Finally, the Syndicator and the Linker can retrieve information about which part of a widget will be used to display a specific Asset field (for example the Header widget part can display the Title Asset field). This relational list of Asset fields corresponding to Widget parts is called Asset Definitions.

3.1.1 First Prototype

The first Format Converter prototype implemented a first approach to a mechanism that can provide HTML documents based on the Asset data it is supplied with. No information on how each specific widget should be handled was embedded in this prototype.

The Graphical Editor was provided in the first prototype with limited editing capabilities of widget styles. The user could create new style templates for any given widget type and the dashboard or edit existing ones. These could be created and edited with the following ways:

- The first way is to use the embedded editor which, in the first prototype, was limited in adjusting the colour values for various parts of a widget or the dashboard
- The other way of editing a style template is to upload a CSS file and thus be able to define anything that can be included in such a file

Finally, all changes could be saved in the cloud.

Apart from the various widget types, the Generic Dashboard editor, part of the Graphical Editor, offered colour configuration only for the header and the main body parts.

The custom CSS files that could be uploaded to the service were automatically applied to the currently selected widget style template. A preview area had also been implemented which displayed the current state of the style template. Finally, the Widget Gallery of the Graphical Editor showed all available style templates stored in the Cloud Storage for a given widget type.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table:

Subcomponent	Task
Format Converter	The base logic of the app has been implemented. Mock Asset data can be converted to a widget according to a mock style.
Graphical Editor	<ul style="list-style-type: none"> • Style template preview (displaying changes in style) • Widget Gallery (displaying all stored style templates) • Custom CSS upload • Colour settings editing • Style export (all changes exported to a single CSS file) • Cloud Storage integration (implemented saving & deleting style templates in the Cloud Storage) • Implemented Generic Dashboard specific style settings

Table 1: Tasks carried out during the First Prototype Implementation

3.1.2 Second Prototype

The Graphical Editor in the second prototype incorporates various editing capabilities of widgets and dashboard styles.

The user can select any widget type or the Dashboard and configure the colour values for specific parts of the widget / Dashboard. Also, the URL of the main image of a widget can be configured. Alternatively, the user can upload a CSS file for more advanced style editing and finally export and the current configuration of a template into a CSS file and download it. A different set of options for colour configuration is available for the Dashboard.

Any style template that is edited can be set as the default one, that is, as the one that will be assigned to the widget each time this widget needs to be displayed. A user can store multiple styles templates in the Cloud Storage, access them through the Template Gallery and set one of them as the default/active one. The Asset Definitions described above can be configured by a dropdown list next to each widget part in the Widget Editor.

The functionality of the Widget can also be configured by uploading HTML code. The uploaded HTML will be first displayed in a Preview pane and upon saving it will be injected into the Dashboard every time it needs to display the specific widget.

A summary of the tasks carried out for each subcomponent of the second version of the prototype is shown in the following table:

Subcomponent	Task
Format Converter	<ul style="list-style-type: none"> • Get HTML (the HTML code for a specific widget is retrieved) • Get Widget types (a list of all the defined widget types is retrieved) • Get Asset definitions (a relation between a specific widget's parts and an Asset's fields is retrieved)
Graphical Editor	<ul style="list-style-type: none"> • Widget HTML code editing • Communication with other SAM component services through TSB (and not directly with them) • Instant preview for all the CSS or HTML changes (instead of having to press a refresh button) • Asset definitions configuration through a drop-down menu next to each widget part • Extension of CSS editing (image configuration) • Improved and more robust network operations • Improved overall User Experience, including better error handling and reporting, a nicer notifications system and more intuitively designed menus and tabs

Figure 6: Tasks carried out during the Second Prototype implementation

3.1.3 Third Prototype

The main addition in the third prototype of both Format Converter and Graphical Editor is the support for accounts. Each content provider can have their own storage place where styles and functionalities can be stored regarding their widgets.

Additionally, a watcher component has been connected with Format Converter which secures that the component will always be functional. Also, Format Converter has been upgraded in order to handle a large number of simultaneous connections in the context of supporting the user trials in September 2016 – see D8.3.2 for more information.

Regarding Graphical Editor, it now contains an HTML Gallery where the user can select stock widgets in order to add them to a specific widget. The ability to add a new custom widget has also been implemented. Now the user has to select among some preloaded and non deletable widgets and any number of user created widgets.

A summary of the tasks carried out for each subcomponent of the third version of the prototype is shown in the following table

Subcomponent	Task
Format Converter	<ul style="list-style-type: none"> • Support for accounts • Mechanism for securing always-on state • Support for large number of simultaneous connections
Graphical Editor	<ul style="list-style-type: none"> • Support for accounts • Increase robustness (extend error handling/logging) • Improvements based on user feedback • HTML template gallery (A gallery where previews to stock HTML codes/widgets will be hosted) • Support for creation of custom widget types

Figure 7: Tasks carried out during the Third Prototype implementation

3.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers.

3.2.1 For Users

The Graphical Editor, being a web application, is intended to be used via any web browser without any special preparations.

3.2.2 For Developers

The Format Converter is written in Python. The Graphical Editor is built with AngularJS and is mostly built with plugins and widgets taken from the common SAM template.

3.3 Installation (Deployment)

Currently, the deployment is carried out by the Jenkins Continuous Integration Server¹ provided by the SAM consortium. For the MDR component, a Jenkins integration project has been created and configured to build and deploy each subcomponent in Apache Tomcat 8.

The Graphical Editor resides in the Administration Tool and the Marketplace while the Format Converter runs as a backend service on the SAM Server.

3.4 Execution and Usage

In the following subsections the execution and usage of the Graphical Editor and Format Converter subcomponents will be explained. The “For Users” section will contain only information about the Graphical Editor, because it is a user interface available through the Administration Tool and the Marketplace. The “For Developers” section will explain the

¹ <http://jenkins-ci.org/>

Format Converter because it is a background service which is available to other components.

3.4.1 For Users

This section describes the steps required for a user to see all available templates for a given widget type (e.g. Facebook), create a new template for it and save it and adjust its colour settings. Also, it is shown how to import and export a CSS file. Regarding HTML, the importing process is explained. Finally, the Asset Definition configuration is explained.

3.4.1.1 See All Available Templates

In order to access this interface, the user needs to be registered within the SAM Platform and logged into the Marketplace. The Widgets option, on the left-hand side menu, provides access to the Graphical Editor, as shown in Figure 8.

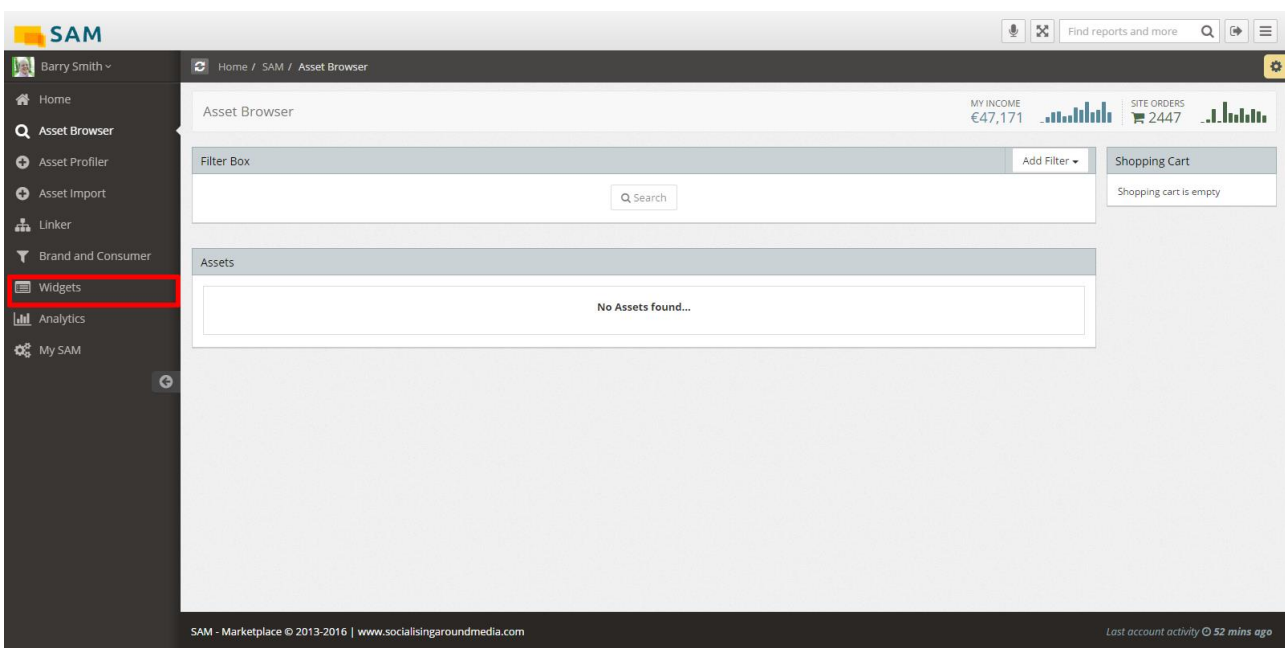


Figure 8: SAM Marketplace Home Screen

In the widgets screen, the dropdown menu at the top right corner allows to select the Widget Type (Figure 9). When the user selects one of the Widget Types (e.g., Facebook), the default style for this widget type is displayed in the HTML Preview, which is below the dropdown menu.

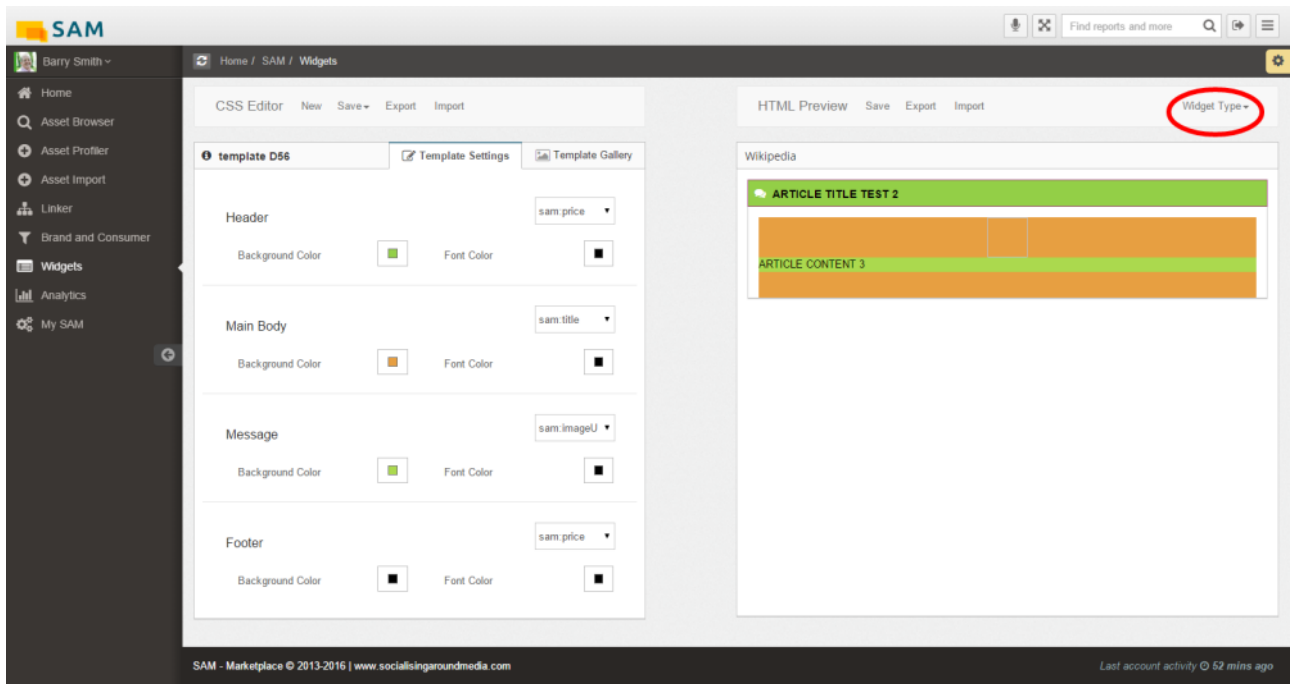


Figure 9: Widgets Component

A user can carry out the following tasks:

- Edit the widget style in the Template Settings tab
- Switch to the Template Gallery tab (Figure 10) and
 - See all available templates
 - Edit, delete or preview (unavailable for the first prototype) a template

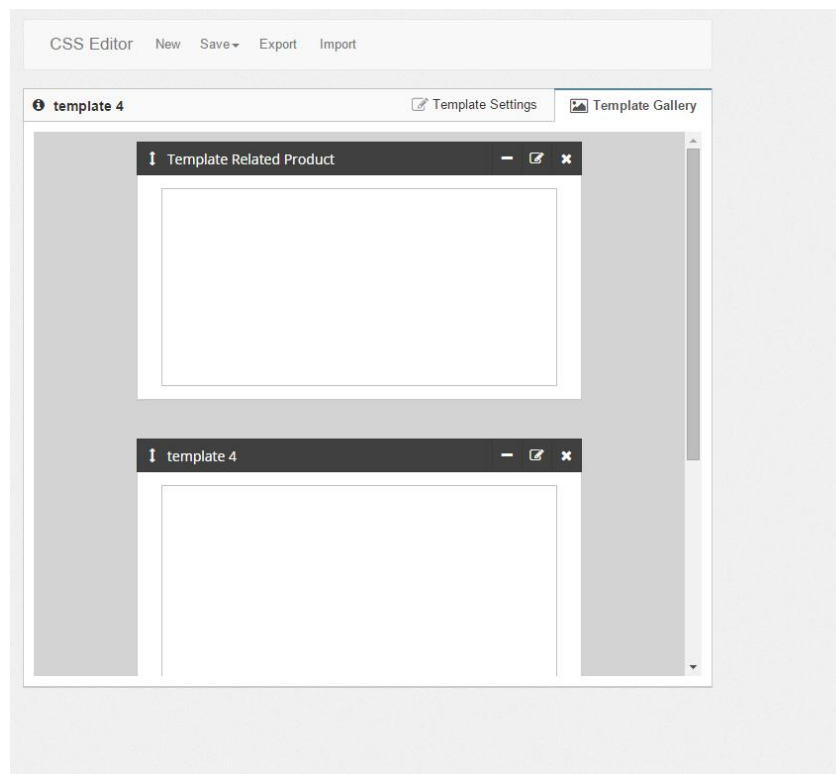


Figure 10: Template Gallery

3.4.1.2 Create a New Style

The user can create a new style by clicking the “New” button. The default style settings will be loaded and the user can select the new configuration. Once the user finishes the configuration, the style can be saved as template. This can be achieved by pressing the “Save” button and selecting the “Save As” sub-option. A popup will request a name and the saving of the template will take place (Figure 11).

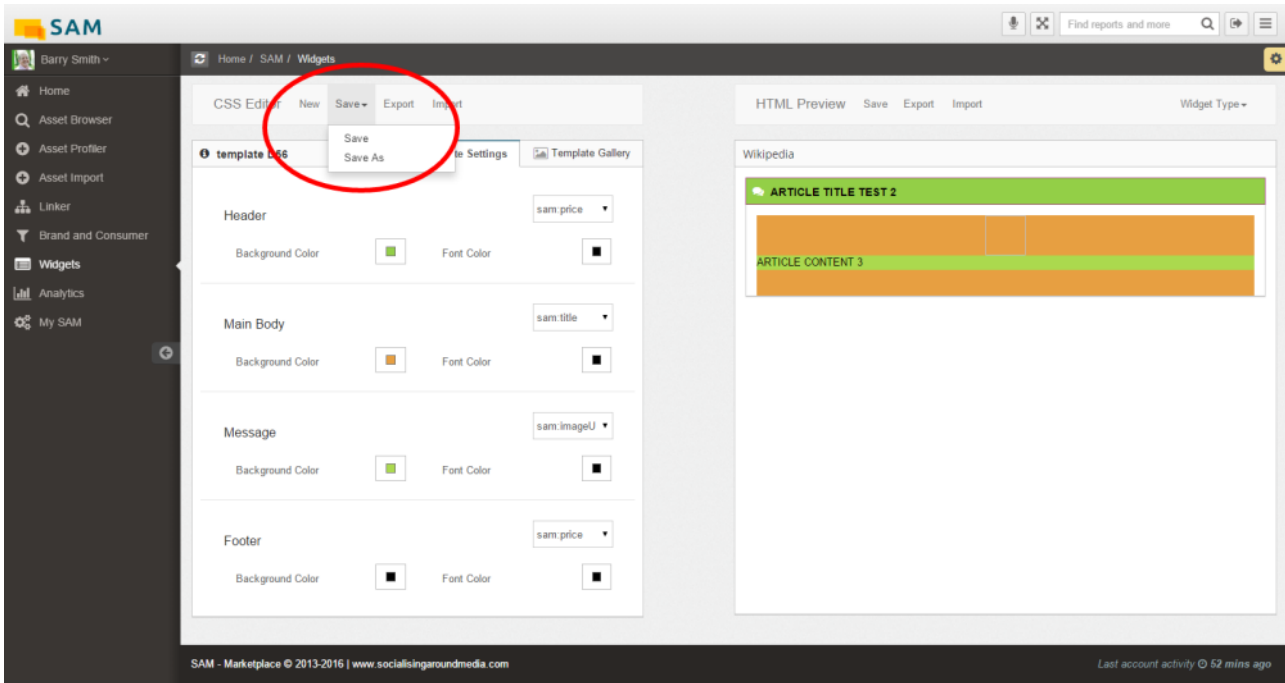


Figure 11: Create New Template and Save it

3.4.1.3 Settings Configuration

For the third prototype, colour and main picture configuration is included in the settings. Also, an Assets Definition dropdown menu is included where the user assigns the specific widget part to an Asset field. All changes made instantly appear on the Preview window.

3.4.1.4 Import/ Export CSS

At any point the user can export the current template configuration (even if it is not saved in the Cloud Storage) to a CSS file. This is done by pressing the Export button (Figure 12).

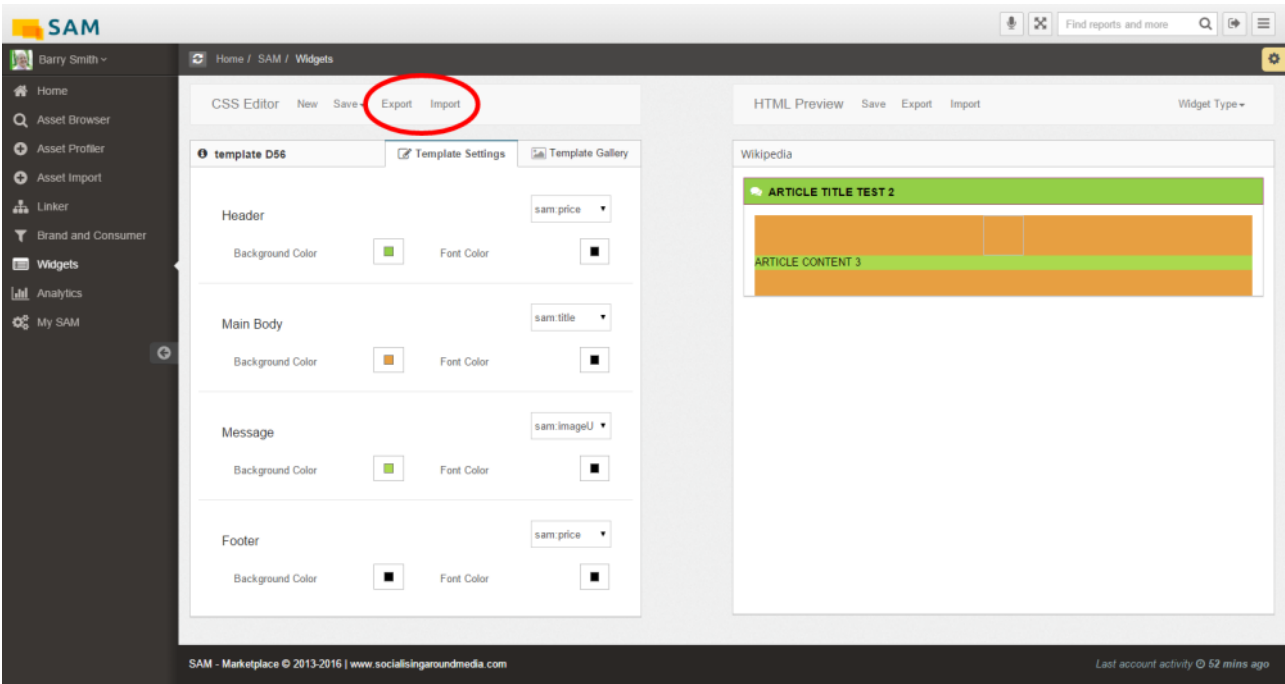


Figure 12: CSS Import/Export

A custom CSS can be uploaded by pressing “Import”. A browser window will pop up and a CSS file can be chosen for uploading (Figure 12). Its content will be directly applied to the Preview area and the configuration settings.

3.4.1.5 Import/Export HTML

The user can update the HTML code of a Widget type and thus define its functionality. The user can update the current HTML by pressing the “Import” (Figure 13). They can also export the current HTML code (even if it is not saved) by pressing the “Export” button (Figure 13).

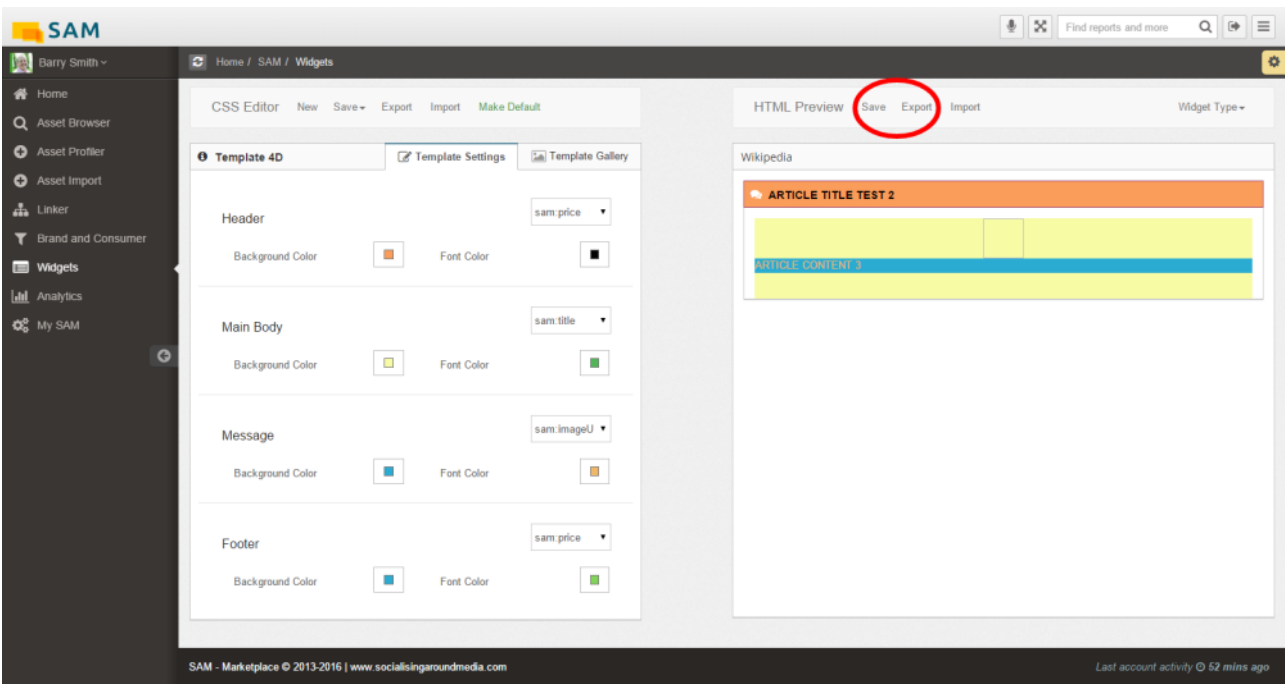


Figure 13: HTML Import/Export

3.4.1.6 Configure Asset Definitions

The Widget Editor also allows for configuration of Asset Definitions meaning the assignment of Asset fields (e.g. title, description) on a specific widget part (e.g. main body, header). This is facilitated by selecting the desired Asset field from the dropdown menu next to each widget part (Figure 14).

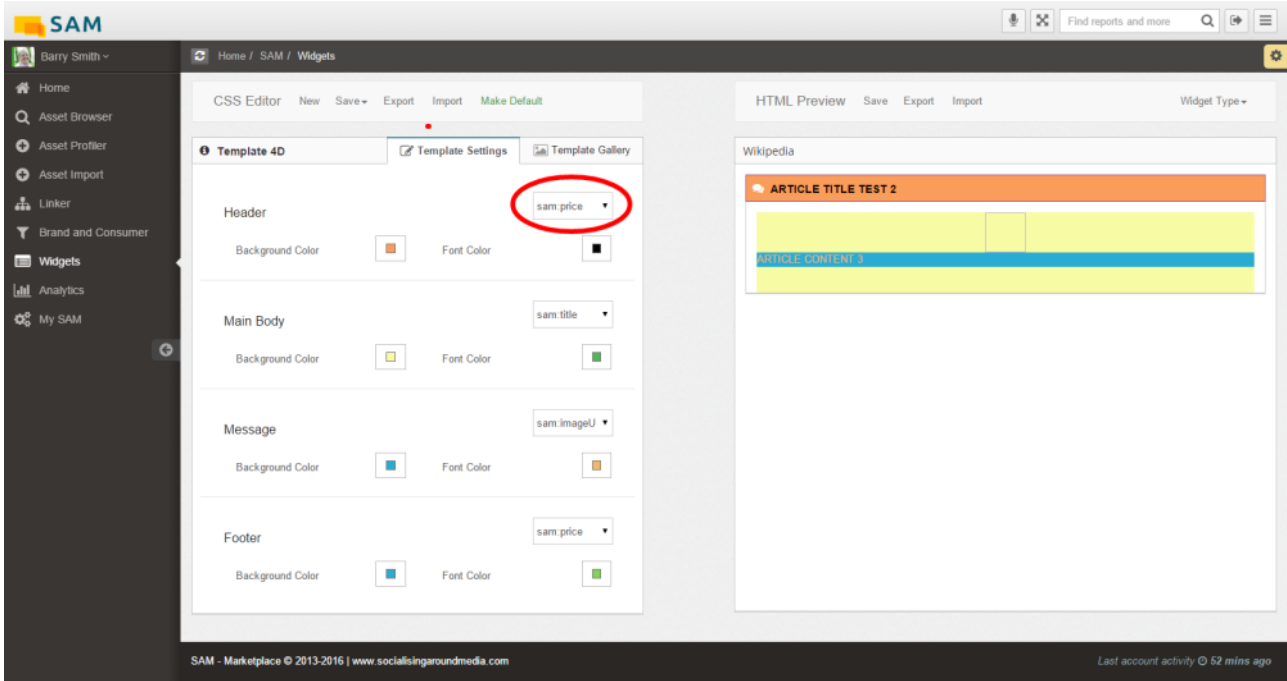


Figure 14: Configuring Asset Definitions

3.4.1.7 Create New Widget Type

In order to create a new widget type the user first has to show the widget list as in Figure 9 and then select the “Manage Widgets” option. Inside the popup the user can add and remove custom widgets. For creating a new custom widget the text input box has to be filled in and then the “Add” button to be pressed.

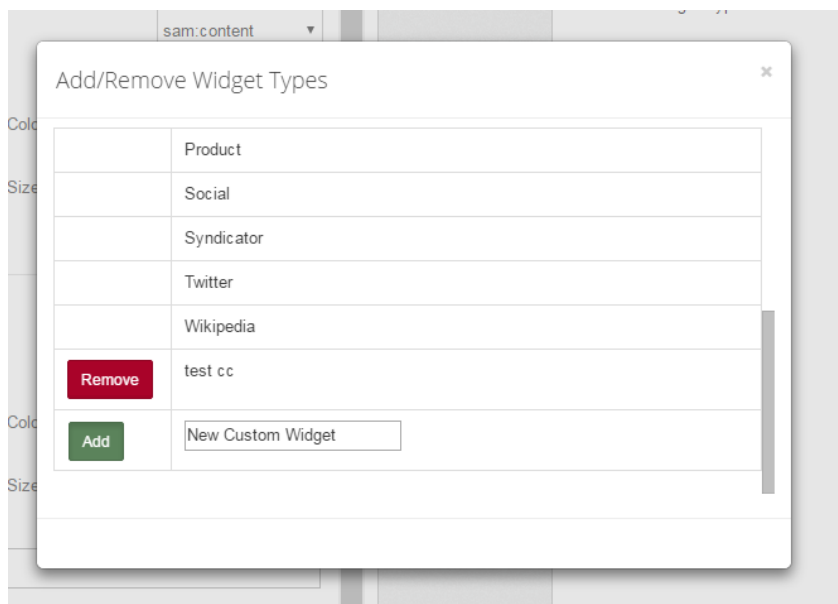


Figure 15: Widget Management Window

After the confirmation that the new custom widget was created, the user can select it from the widget menu and start editing its style and functionalities.

3.4.1.8 Use Stock HTML Templates

The Graphical Editor sports a variety of pre-made HTML templates that the users can apply to a widget. To do this, they need to select the “HTML Gallery” tab and then click on “Import” button on any template they would like to apply.

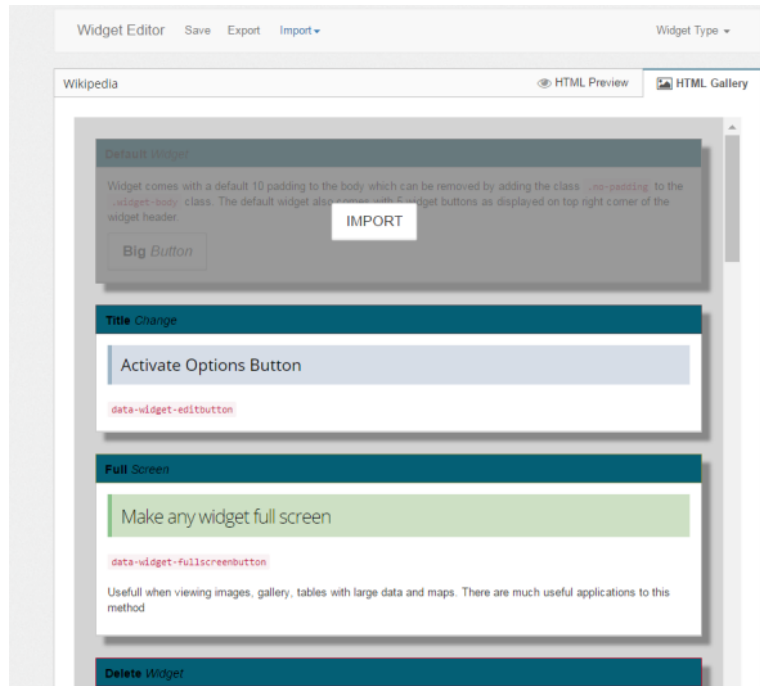


Figure 16: HTML Gallery

Notice that the template is not saved yet. The “Save” button has to be pressed for that.

3.4.2 For Developers

The Graphical Editor is a graphical interface and thus not programmatically available. The Format Converter operates as a REST service available for other subcomponents. It can be queried with the REST methods described below.

3.4.2.1 Request CSS

Request CSS		
Description	Requests a CSS string with the default style for a specific widget type	
Request		
Request URL	POST http://localhost:9098/get_css	
HTTP Parameters	account Required string	ID of the account that the CSS data come from Example: “BDS”
	widget_type_id Required String	ID of the Widget type that the CSS will be about Example: “Wikipedia”

Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
Body	String	A string containing the requested CSSA s

Figure 17: Request CSS Method

3.4.2.2 Request Widget List

Request Widget List		
Description	Requests a list with all the widgets in use	
Request		
Request URL	POST http://localhost:9098/get_widget_types	
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
JSON Attributes	JSON Object	An array containing IDs of all widget types in use

Figure 18: Request Widget List Method

3.4.2.3 Request HTML

Request HTML		
Description	Requests an HTML string containing the code of a specific widget type	
Request		
Request URL	POST http://localhost:9098/get_html	
HTTP Parameters	account Required string	ID of the account that the HTML data come from Example: "BDS"
	html_id Required String	ID of the Widget type that the HTML will be about Example: "Wikipedia"
Response		
HTTP Status	Value	Description

Code	200	Object found
	204	Object not found
	400	Bad Request
Body	String	A string containing the requested HTML

Figure 19: Request HTML Method

3.4.2.4 Request Asset Definitions

Request Asset Definitions		
Description	Requests all the Asset definitions for a given widget type	
Request		
Request URL	POST http://localhost:9098/get_asset_definitions	
HTTP Parameters	account Required string	ID of the account that the asset definitions come from Example: "BDS"
	widget_type_id Required String	ID of the Widget type that the asset definitions will be about Example: "Wikipedia"
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
Body	JSON Object	An object containing the asset definitions

Figure 20: Request Asset Definitions Method

3.4.2.5 Request non CSS Data

Request Non CSS Data		
Description	Requests a JSON object with all non CSS data of a style (e.g. image URLs)	
Request		
Request URL	POST http://localhost:9098/get_non_css	
HTTP Parameters	account Required string	ID of the account that the non CSS data come from Example: "BDS"
	widget_type_id	ID of the Widget type that the non CSS will be

	Required String	about Example: “ <i>Wikipedia</i> ”
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
Body	JSON object	A JSON object containing the requested non CSS data

Figure 21: Request CSS Method

3.4.2.6 Update HTML

Update HTML		
Description	Submit HTML code for a specific widget type	
Request		
Request URL	POST http://localhost:9098/update_html	
HTTP Parameters	account Required string	ID of the account that the widget type will be edited of Example: “BDS”
	widget_type_id Required String	ID of the widget type that will be edited Example: “ <i>Wikipedia</i> ”
Body	String	HTML code
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request

Figure 22 Update HTML Method

3.4.2.7 Delete Widget

Delete Widget	
Description	Delete specific widget type
Request	
Request URL	POST http://localhost:9098/delete_widget

HTTP Parameters	account Required string	ID of the account that the widget type will be deleted of Example: "BDS"
	widget_type_id Required String	ID of the widget type that will be deleted Example: " <i>Wikipedia</i> "
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request

Figure 23 Delete Widget Method

3.4.2.8 Create Widget

Create Widget		
Description	Create a new widget type	
Request		
Request URL	POST http://localhost:9098/create_widget	
HTTP Parameters	account Required string	ID of the account that the widget type will be deleted of Example: "BDS"
	widget_type_id Required String	ID of the widget type that will be created Example: " <i>Wikipedia</i> "
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request

Figure 24 Create Widget Method

3.5 Limitations

In case that the Graphical Editor becomes a product, an HTML validation tool should be implemented so that the users get automatically and immediately notified whenever an HTML is not valid. Also, snapshots of the different styles in the CSS gallery would be useful so that the user does not need to preview them.

Regarding Format Converter, advanced feedback should make the component easier for developers. By advanced feedback it is meant a more detailed and specialised error report when the return value is not the desired one.

3.6 Research Background

The following figure provides the research publications that helped to design Format Converter and the Graphical Editor. Apart from that, considerable time was dedicated to experiment with various techniques in order to tackle specific obstacles in the development of the Graphical Editor. This experimentation consisted of reading official documentations of technologies in use and eventually adjusting code appropriately. In cases where this did not have the expected results, alternative frameworks were experimentally applied and where the results were satisfactory these frameworks were integrated in the code.

Source	Subcomponent	Description
[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.	Format Converter & Graphical Editor	Describes the basic principles and philosophy behind using widgets as the primary place of End User interaction.
[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development	Format Converter & Graphical Editor	This poster describes the experiences of Science Gateways developers of using Twitter Bootstrap and AngularJS frameworks in order to address the balance between design and implementation. It is described how using these tools eventually empowered them to create better styled and easily maintainable websites.
[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets	Format Converter & Graphical Editor	Mobile widgets provide an elegant way of delivering personalised web content and especially Web 2.0 services to mobile devices. This paper introduces the concept of widgets and general principles of widget development. The main section deals with the characteristics of mobile widgets and outlines differences to desktop widgets and traditional mobile application development platforms.

Figure 25 Consulted Research Publications

3.7 Target Performance

This section contains the key performance indicators (KPI) (see Section 3.7.1) and user experience measurement tasks (see Section 3.7.2) for this component.

3.7.1 Component KPIs

The Key Performance Indicators (KPIs) for this component are defined in Figure 26. The results vary depending on the indicator.

Topic	Description	Target KPI
Responsiveness	When a user is interacting with the Graphical Editor, it needs to be responsive.	Initial loading time of the Graphical Editor should be less than 5 sec. Editing actions which include cloud storage communication should be executed in less than 2 sec.
Ease of use	The ease of use is an important aspect of the Graphical Editor. Users use its UI in order to style widgets, configure their behaviour and organize these configurations in the Cloud Storage. A user-friendly environment is a necessity in order to take advantage of all these include functionalities.	Based on a short user test, the following metrics are taken under consideration. <ul style="list-style-type: none"> • Completion rate • Usability problems • Task time
Accessibility	The Graphical Editor should work in the most important browsers and it should be well formed HTML and CSS mark-up.	Graphical Editor should work in Chrome and Firefox.

Figure 26: Target Performance MDR

Regarding the Responsiveness indicator, the Graphical Editor meets all requirements excluding the fetching of Asset Characterisations which occasionally may exceed the 2 seconds limit.

Regarding the Ease of use, a short user trial was arranged in order to test the Graphical Editor (see 3.7.1.1).

Occasional uses of the Graphical Editor with a browser other than the development one (Chrome) did not show any problems.

3.7.1.1 User Trials

In order to formally test the overall design and functionality of the Graphical Editor and to complement the feedback given from the various Production Trials a small testing session was organized in the offices of Talkamatic in Gothenburg, Sweden.

The participants were four and all employees of Talkamatic. The main objective was to measure the time needed for the testers to complete the tasks given to them. The participants' acquaintance with using computer software ranges from extended to professional. No participant had previous experience of using SAM Marketplace and the Graphical Editor in particular.

The participants were given a list of scenarios (see Graphical Editor User Trials Scenarios) and were asked to complete one at a time.

All participants were able to complete all tasks. No task took more than one minute for the user to complete, with most of them taking less than a half a minute. Overall, the measurements can be characterised as very satisfactory, especially taking into consideration that the concept of the SAM Marketplace had to be explained to them briefly before execution of the scenarios.

3.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 3.7.1, this work package provides user tasks, which form an input for measuring the subjective user experience in a uniform way. For each of the tasks below, the task-specific KPIs defined in following Annex A: User Research will be measured.

Task	Description
Customize default template	The user must find the default template of a specific widget type she wants to edit and then perform a number of alterations to the template's settings.
Manage templates	The user must promote another template to the default status, delete a template, create a new one and rename one.
Import and export template	The user must import and export templates.

Figure 27: Target Performance MDR – Task-Specific Measurements

3.8 Summary

This section provides a description of the third prototype of the Multi Device Representation component developed in task T7.1 Multi-Device Content and Media Representation. The main outcomes of this task are two pieces of software: the Graphical Editor and the Format Converter. This prototype is the third of the three iterations planned for this component.

The prerequisites necessary for both users and developers to install and use the Graphical Editor and Format Converter have been described.

Regarding the Format Converter, services for retrieving styles and code from any widget type have been implemented. For the Graphical Editor, the component is now using TSB, supports HTML editing and has improved overall user experience.

The last section has been dedicated to describing the performance and limitations of the third version of the component, which is delivered in M37.

4 SAM 1st Screen Media Interaction

This section describes the software deliverable D.7.3.3, which is the third prototype release of the SAM 1st Screen Media Interaction functionalities.

4.1 Scope and Relationship

The 1st Screen component embeds the Generic Dashboard and enables video streaming for the End User view. The End User can in some cases also consume related content displayed inside widgets on the 1st Screen.

Figure 28 shows the different subcomponents of the 1st Screen component, the logical connections that have been established between them and the relations with other components and actors in the SAM Platform.

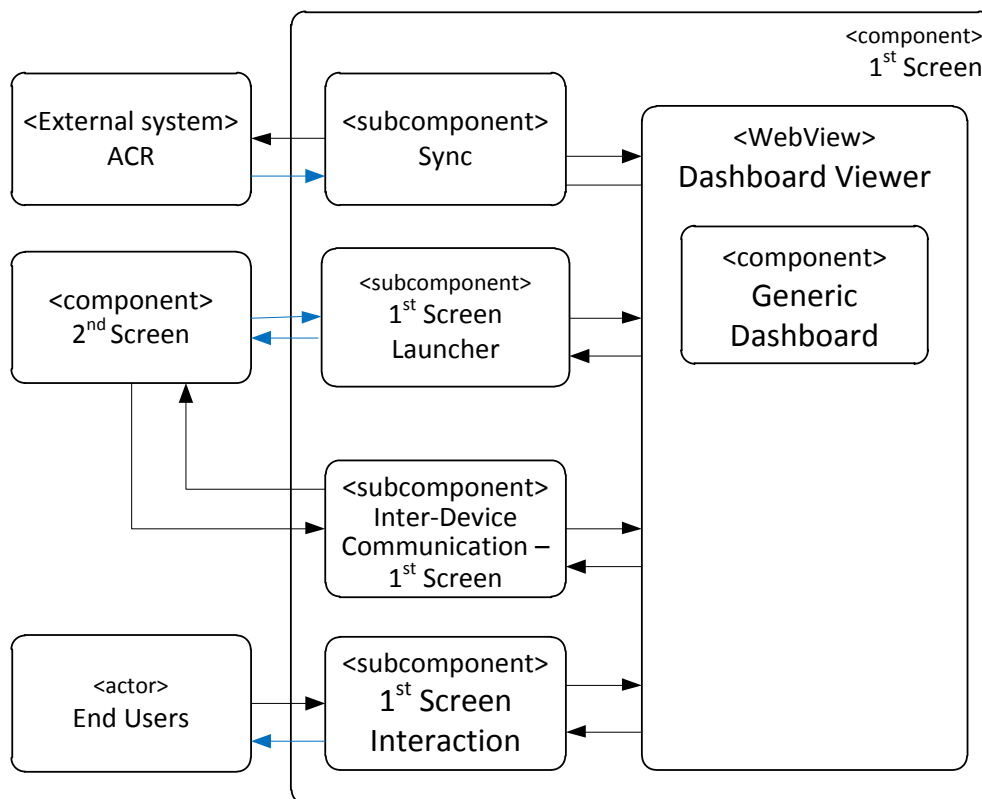


Figure 28: 1st Screen Subcomponents and its Relationships

For further descriptions of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.13.3 (Architecture), D3.2.2 Section 4.14.2 (Functional Specification) and D3.3.1 Section 3.14.2 (Technical Specification).

The first prototype of T7.3 provided the basic integration of the Generic Dashboard component using the Dashboard Viewer subcomponent, which provided the display. Basic remote control functions had also been implemented using 1st Screen interaction component, which allowed the End User of the 1st Screen application to navigate through the screens, select a video stream, start/pause/stop the video and toggle full screen mode.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table:

Subcomponent	Task
1 st Screen Interaction	Remote control support.
Inter-Device-Communication 1 st Screen	Simple communication between 1 st and 2 nd Screen devices using centralised Web Socket server.
Dashboard Viewer	Display of the Generic Dashboard component. Provide 1 st Screen device specifications.
Sync	No tasks planned for the first prototype.

Figure 29: Tasks carried out for the First Prototype of T7.3

The second prototype of T7.3 provided further integration of the Generic Dashboard component using the Dashboard Viewer subcomponent, which provides the display. The embedded Web Socket server as a part of 1st Screen Inter-Device Communication was also implemented, which enables more efficient local communication between 1st and 2nd Screen devices. Furthermore, to add the possibility of automatic 1st Screen discovery, an Android NSD protocol implementation was put in place.

A summary of the tasks carried out by each subcomponent of the second version of the prototype is shown in the following table:

Subcomponent	Task
1 st Screen Interaction	No tasks planned for the second prototype.
Inter-Device-Communication 1 st Screen	<ul style="list-style-type: none"> Implementation of local Web Socket Server in the 1st Screen allowing automatic synchronization between 1st Screen and 2nd Screens over a local network. Advertisement of the 1st Screen application using Android NSD protocol.
Dashboard Viewer	<ul style="list-style-type: none"> Implementation of local HTTP Web Server in the 1st Screen allowing for local storage and access to Generic Dashboard.
Sync	No tasks planned for the second prototype.

Figure 30: Tasks carried out for the Second Prototype of T7.3

The third prototype of T7.3 provides the further optimization of the Generic Dashboard communication with other involved components such as 1st and 2nd Screen and Syndicator.

Implementation of Android NSD protocol was improved allowing for constant discovery and reconnection in case of connectivity problems.

Servers were further optimised which resulted in faster loading times of the 1st Screen app. Caching mechanism was improved to handle different possible use cases.

Several fixes were introduced to improve user experience and to make application more robust with respect to connection handling (e.g. properly cleaning up connections to 2nd Screen clients when 1st Screen or 2nd Screen application is stopped).

A summary of the tasks carried out by each subcomponent of the third version of the prototype is shown in the following table:

Subcomponent	Task
1 st Screen Interaction	No tasks planned for the third prototype.
Inter-Device-Communication 1 st Screen	<ul style="list-style-type: none"> • Further optimization of web sockets and web server on 1st Screen. • NSD reconnection.
Dashboard Viewer	<ul style="list-style-type: none"> • Proper handling of widgets caching mechanism in multiple 2nd Screen users' scenario.
Sync	No tasks planned for the third prototype.

Figure 31: Tasks carried out for the Third Prototype of T7.3

4.1.1 1st and 2nd Screen Interaction

To facilitate discovery and interaction between 1st Screen and 2nd Screen, some standard protocols need to be implemented. For SAM it has been decided to use Network Service Discovery (NSD) for discovery mechanism between 1st Screen and 2nd Screens (See section 4.1.1.1). Also a design decision was made to implement Web Socket Server on 1st Screen to host Generic Dashboard application (See section 4.1.1.2). This decision was made to improve the performance of the system by host the application in local network rather than running on remote network (See section 4.1.1.3).

4.1.1.1 Network Service Discovery (NSD)

NSD² is a simple protocol, which added to an app allows the users of it to discover other devices on the local network that support the services the app requests. This is useful for a variety of peer-to-peer applications such as communication between 2nd Screen and 1st Screen for exchange of information. Android's NSD API simplifies the effort required to implement such features.

In the third prototype, a major rework of NSD implementation was introduced. To properly detect connectivity issues, the discovery is always active both on 1st and 2nd Screen and each time one of the two becomes unreachable, a proper call-back notifies the other about the issue, allowing a proper clean-up of network resources.

4.1.1.2 Web Socket Server

A design decision was made to move the Web Socket Server implementation from Generic Dashboard server to 1st Screen application on the TV. Web Socket server on the TV is triggered from inside the 1st Screen application when the application is started.

The Web Socket server and client implementations are written in Java 1.7 version. The underlying classes are implemented using *java.nio*, which allows for a non-blocking event-driven model (similar to the Web Socket API for web browsers).

Implemented Web Socket protocol versions are:

- RFC 6455³
- Hybi 17⁴
- Hybi 10⁵
- Hixie 76⁶

² <http://developer.android.com/training/connect-devices-wirelessly/nsd.html>

³ <http://tools.ietf.org/html/rfc6455>

⁴ <http://tools.ietf.org/id/draft-ietf-hybi-thewebsocketprotocol-17.txt>

⁵ <http://tools.ietf.org/id/draft-ietf-hybi-thewebsocketprotocol-10.txt>

- Hixie 75⁷

The *org.java_websocket.server.WebSocketServer* abstract class implements the server-side of the Web Socket Protocol⁸. A Web Socket server by itself does not do anything except establish socket connections through HTTP. After that it's up to the implementing subclass to add business logic.

The *org.java_websocket.client.WebSocketClient* abstract class can connect to valid Web Socket servers. The constructor expects a valid *ws://* URI to connect to. Important events *onOpen*, *onClose*, *onMessage* and *onIOError* get fired throughout the life of the Web Socket Client, and must be implemented in the relevant subclass.

4.1.1.3 Web Server and Hosting Generic Dashboard in 1st Screen

A design decision was made to move Generic Dashboard hosting from the remote server to the 1st Screen application on TV.

To host the files and related resources, a webserver is needed to be in place. The webserver needs to take care of following functionalities:

- Host source code files and related resource files, images, icons, etc.
- Process clients' requests made by 2nd Screen applications and feed them processed data accordingly

There are different variants of webserver based of the type of protocol used for implementation. As the Generic Dashboard was previously hosted on a HTTP based server, it was decided to create the same variant of web server using HTTP protocol in the 1st Screen application. With this in place, all the components which need to interact with the webserver continue to work in the same way without any modification required on their side.

4.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers.

4.2.1 For Users

The user needs to provide an Android Smart TV device running Android version 5.1 and supporting SmartTV specification version 3.0⁹. The Android application is not available on Google Play¹⁰. Due to the fact that Philips 2k15 TV's do not allow apps coming from other sources than the Google Play Store or the Philips SmartTV Portal, the *.apk file can only be installed on a Philips Development TV model. On these models the app can be installed through the usage of android debug bridge (where x.x.x.x is the IP address of the TV):

```
adb connect x.x.x.x:5555
adb install 1st-screen.apk
```

⁶ <http://tools.ietf.org/id/draft-hixie-thewebsocketprotocol-76.txt>

⁷ <http://tools.ietf.org/id/draft-hixie-thewebsocketprotocol-75.txt>

⁸ <http://www.whatwg.org/specs/web-socket-protocol/>

⁹ <https://developers.smarttv-alliance.org/specification>

¹⁰ <http://play.google.com/>

4.2.2 For Developers

For developers it is strongly recommended to use the Android Studio IDE¹¹ for the implementation. Going forward, Google has deprecated the support for Eclipse for Android app development which was used before and is pushing all to migrate to Android Studio IDE. Also, Android Studio IDE has much better support for Android application development.

On the other hand, Java-WebSocket is known to work with:

- Java 1.7 (SE 7)
- Android 1.6 (API 4)

Some other Java versions may also be used for this implementation but these have not been tested.

To create a typical webserver, following steps are required:

1. Initially a server socket needs to be created which should listen to the desired port. To create a server socket the *java.net.ServerSocket* class is used. The constructor of this class accepts a port number to listen for the incoming connections.
2. Once the *ServerSocket* object is created, it is needed to accept the incoming connection using *ServerSocket.accept()* method. This method is blocking, so a separate thread is created to accept and process the incoming connections. The *accept()* method returns a *java.net.Socket* object which represents the accepted connection.
3. Once the connection is established, the next HTTP request needs to be processed. This can be done using *org.apache.http.protocol.HttpService* class. This class provides a server side implementation which can handle minimal HTTP processing requests. In the SAM implementation, to handle different HTTP requests, handler map design is used which is based on URI patterns.

An implementation of a local webserver to serve Generic Dashboard directly from the local network was decided to be part of the third prototype, but has been implemented in the second prototype. This decision was made to improve the performance of the system and to make a better demonstrable product in places with weak Internet connections.

4.3 Installation (Deployment)

This section describes the Installation and deployment process to use the app and source code for Users and Developers.

4.3.1 For Users

After downloading the 1st Screen application and moving it to the device's internal or external storage, the user has to execute the *.apk file. As the first step, the user will be prompted for the acceptance of the access rights. After the installation, the 1st Screen application can be started using the icon in the app overview.

¹¹ <http://developer.android.com/tools/studio/index.html>

4.3.2 For Developers

Developers have to download the source code first (Deliverable D7.3.3 contains the required information). The source code of the 1st Screen application is an Android Studio project and can be opened with Android Studio after downloading.

4.4 Execution and Usage

This section describes how to use the different subcomponents of the prototype.

4.4.1 For Users

The following subsections present the different views of the 1st Screen application.

4.4.1.1 Generic Dashboard Viewer

When the application is started, it will launch a SmartTV capable browser to view the Generic Dashboard. Depending on the network bandwidth this can take a few seconds during the first start-up of the application. The home screen of the Generic Dashboard is shown in Figure 32.

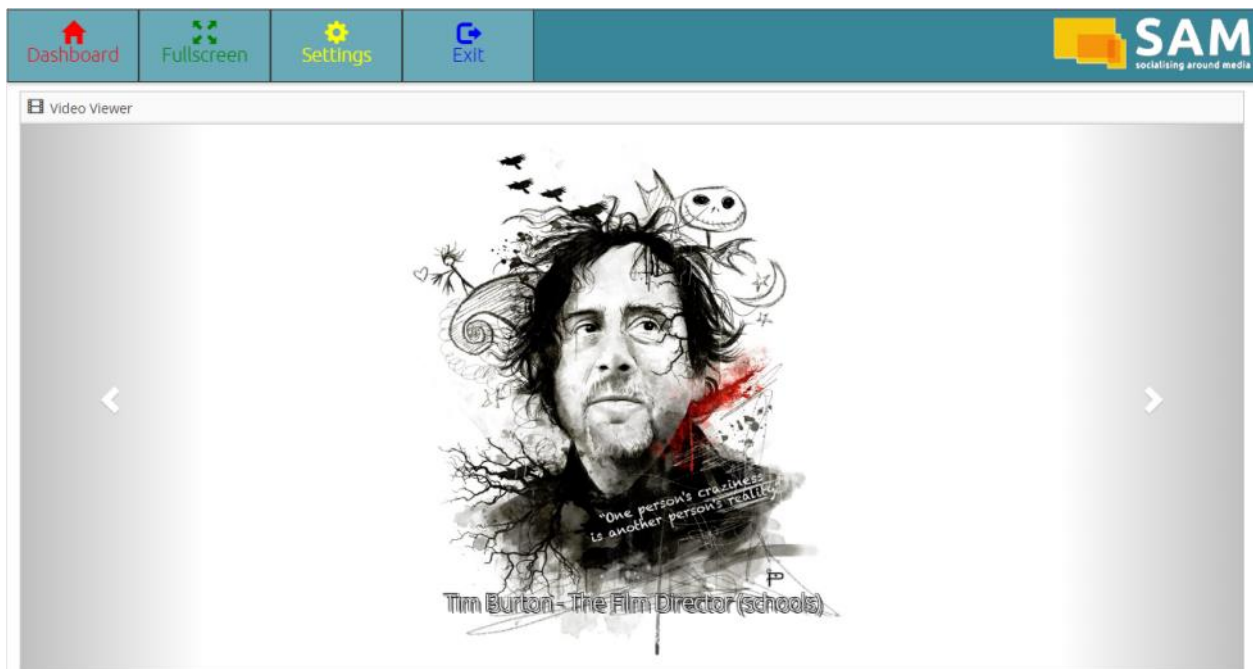


Figure 32: 1st Screen Application Generic Dashboard View

The home screen consists of four navigation buttons and three Widgets. The End User can employ her/his remote control to navigate between the buttons and Widgets using remote's navigation buttons (up / down/ left / right). When the End User navigates the Generic Dashboard, a blue cursor frame is visible to indicate currently focused element. Pressing OK will execute requested action. Colour keys on the remote can also be used to quickly select access the functionality behind the four navigation buttons:






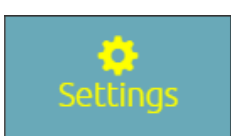


Remote button	On screen button	Function
		View the home screen of Generic Dashboard
		Toggle full screen video player
		View SAM settings page
		Exit the 1 st Screen application

Figure 33: Generic Dashboard Home Screen

The Widgets presented on the Generic Dashboard are: Video Viewer, Device Info and Video Info. The Device Information Widget displays, amongst other information, the display size and density of the device and its GPS position (if known). Video Viewer displays currently available video streams for End User to browse and view. Once a selection has been made, the video is buffered and then played. The End User can use the play/pause/stop buttons on the remote to control the player. Video Info Widget contains the stream name and current position of streamed video.

4.4.1.2 SAM Settings Page

After selecting the Settings button (or pressing yellow key on the remote), the End User is presented with SAM settings page (see Figure 34). On this page, the user can toggle the state of various SAM-related settings (e.g. advertising, statistics, geo-location).

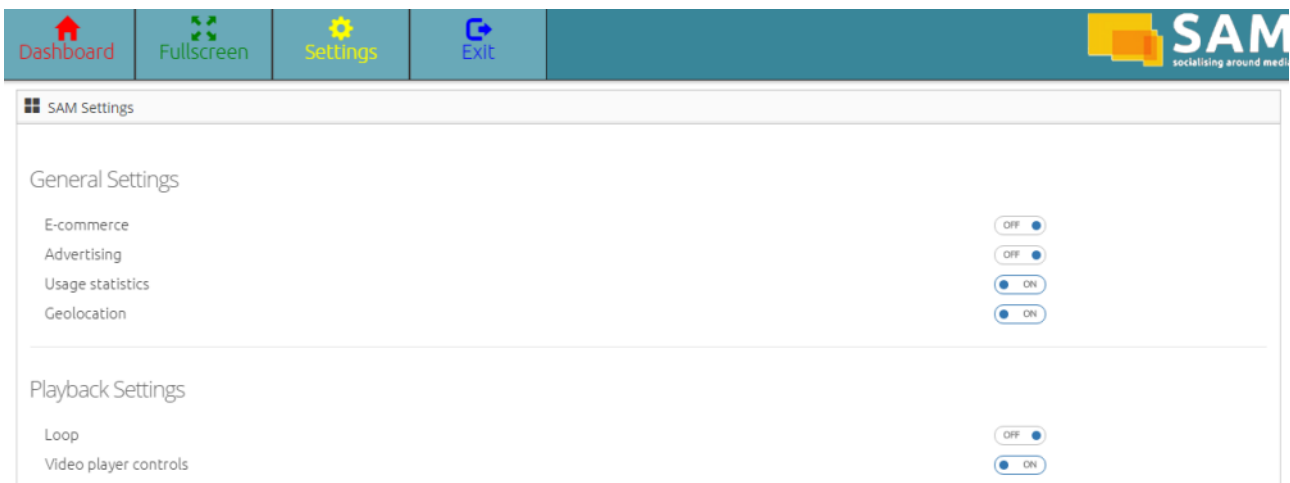


Figure 34: SAM Settings Page of 1st Screen Application

Due to the SmartTV application nature, the rest of 1st Screen components have been included in Generic Dashboard source code. Please, for further details refer to the Generic Dashboard Section in document D7.9.4 **Error! Reference source not found.**

4.4.2 For Developers

Once the source code is downloaded and opened using Android Studio, the application can be compiled and executed using Android Studio only. Android Studio has built-in tools to support compilation and installation of the app on to the device. Also, Android Studio has built-in tools which support debugging the application.

4.5 Limitations

Development of the third prototype has been concentrating on bringing the last improvements to the 1st Screen Android application. Due to that, the following view/function has been provided as mock-up:

- **1st Screen Launcher:** Currently the third prototype does not include any remote functionality for 1st Screen Launcher, thus it is needed to manually start the 1st Screen app before connecting any of the 2nd Screen devices.

4.5.1 Possible further improvements

Given that this project delivers a working prototype and not market ready application, the following things can be considered when moving this prototype to further stages towards the market:

- **Dashboard branding:** From business point of view introducing dashboard themes in brand colours and logos could be a key in bringing this app to market.
- **New 1st screen widgets:** 1st Screen widgets are limited currently to those providing optimal viewing experience of provided content. Furthermore other widgets (similar to those on 2nd Screen) could be introduced to 1st Screen to enrich this experience.

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
[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on , vol., no., pp.1,5, 9-11 Sept. 2013	Inter-Device-Communication 1 st Screen	This paper has been considered to implement communication protocols according to the HbbTV standard.
[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).	Inter-Device-Communication 1 st Screen	Input on 2 nd Screen users as background information on profiling.
[VAN14]] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International	Inter-Device-Communication 1 st Screen	Highlights results from EU project Hbb-NEXT, in particular applied to

Workshop on Interactive Content Consumption, Newcastle upon Tyne		recommenders. See http://www.hbb-next.eu/index.php/documents
--	--	---

Figure 35 Consulted Research References

4.7 Target Performance

The performance measurement of the 1st Screen prototype will be measured according to the defined KPIs (described in Section 4.7.1) with additional End User experience measurements (described in Section 4.7.2).

4.7.1 Component KPIs

For this component the following Key Performance Indicators (KPIs) have been defined:

Topic	Description	Target KPI	Current KPI
Responsiveness	When a user is interacting with his 1 st Screen device, it needs to be very responsive.	Initial loading time of the 1 st Screen app should be less than 10 sec. Navigation actions should execute in less than 3 sec.	Current loading time of 1 st Screen app is around 5-8 secs. Navigation actions execute in less than 2 secs.
Connection success rate	When a 1 st Screen discovers a 2 nd Screen (or vice versa), setting up a connection should always be successful.	The discovery of all available 2 nd Screen devices should be achieved in 90% of the test cases.	The discovery succeeds in 90% test cases.
Connection setup time	The time it takes to setup a connection between a 1 st Screen and a 2 nd Screen device needs to be low.	The discovery of all available 2 nd Screen devices should be achieved in less than 10 seconds.	The discovery time is around 5-7 secs.

Figure 36: Target Performance 1st Screen

4.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 4.7.1, this work package provides user tasks, which are input for measuring the subjective user experience in a uniform way. For each of the tasks below, the task-specific KPIs defined in Annex A: User Research will be measured.

Task	Description
Pairing	The user has to pair the 1 st Screen device with the 2 nd Screen device.
Controlling the Video Element	The user has to stop, to shift to a specific time in the video element and start the video element again.
Making video full screen	The user has to make the video full screen and later get back to the non-full screen mode again.

Figure 37: Target Performance 1st Screen – Task-Specific Measurements

4.8 Summary

This section provides a description of the third prototype of the 1st Screen component developed in task T7.3 1st Screen Media Interaction. The main outcome of this task is the software of the 1st Screen component. This prototype is the last iteration planned for this component and the goal of this prototype is to cover 100% of the requirements of the component (see Section B 1.3.3.7 of the DOW for additional information on the effort distribution for this component in the lifespan of the project).

The most important goal reached during this third prototype has been the introduction of a more lenient user experience by further extension of 1st Screen communication protocols.

The requirements necessary for both users and developers have been presented including installation instructions. The last section has been dedicated to describing the remaining limitations of the current prototype and potential further improvements when transforming into a market product.

5 Document Summary

This document, D7.9.3, is the third release of the deliverable series D7.9.x to provide early insight of the prototypes of software deliverables D7.1.3, D7.2.3, D7.3.3 and D7.4.3. This document provided information about the running tasks in Work Package 7:

- T7.1 Multi-Device Content and Media Representation (Section 3)
- T7.3 1st Screen Media Interaction (Section 4)

The information for each task provided contains:

- Scope of the pilot implementation, its purpose and the main relationships with other modules implemented in SAM in the third year of development
- Information needed to deal with the pilot in terms of technical and non-technical requirements, software to be installed, etc.
- Steps needed to install the pilot software and process to build it from source code
- Different screens and actions implemented at the pilot itself, ways to access it, and to test the different implemented options
- Current pilot limitations and the expected improvements
- Papers and other scientific information considered
- Key Performance Indicators (KPI's) for the SAM component
- Conclusion of the implementation of the first, second and third prototype

References

- [WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.
- [BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development
- [KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets
- [LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic
- [BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005
- [BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.
- [ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013
- [VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).
- [VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne
- [BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013
- [ETS15] ETSI TS 102 796 V1.3.1 / HbbTV 2.0 (2015-10)
- [WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.
- [MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research

Generic Target Performance KPIs

For assessing the target performance of different components of the Multi-Device Media Representation and Interaction work package (Sections 3 and 4), five Target KPIs were defined as described below. Every task defined in sections 3.7.2 and 4.7.2 will be used to measure the three task-specific target KPIs. Next to that, two general usability KPIs were defined in order to measure the overall user experience.

Type	Type	Description	Target KPI
Task-specific	First Click score	After a limited amount of usage (familiarisation time, described below), did the user click the correct button for the optimal path to complete the requested task?	80% correctness for “top tasks” should be reached, 65% for other tasks.
Task-specific	Task Completion	Was the user able to complete the task within a reasonable time? (A reasonable threshold depends on the task and on the product – it’s best for someone who knows it well to set this).	80% correctness for “top tasks” should be reached, 65% for other tasks.
Task-specific	Task Confidence	How confident is the user that s/he completed the task requested?	Average score of 5 or above (7-point scale).
Overall Usability	System Usability Scale (SUS) ¹²	This is a survey with 11 statements with an associated 7-point scale. The procedure for executing the SUS is described on the website in footnote 12.	Composite of 70 or higher (100-point scale).
Overall Usability	Usability Adjective Scale	This is a single question that is usually added to the end of the SUS that gives a scale of adjectives to describe whatever system is being measured from “worst imaginable” to “best imaginable”.	4 or higher (7-point scale).

Figure 38: Generic Target Performance KPIs

Testing Procedure

A very short description of the testing procedure to follow is described below. The details can be found on the website in footnote 12.

1. Recruit at least 10 research participants with little to no knowledge of the system.
2. Give each of these participants a brief (less than 5 minute) introduction to the feature under test. If possible, use the actual materials that a consumer would receive.
3. Foresee for each participant a short amount of time to familiarise himself with the feature (usually less than 5 minutes for a consumer is good).
4. The moderator then asks each participant to complete a series of tasks. These tasks should be the most common actions users would take when working with the feature. (For SAM, this would include widget configuration, TV-side navigation, and 2nd Screen app usage.) Within each of those tasks there may be more than one sub-task for the

¹² <http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>

participant. Of these sub-tasks, identify, by observing the behaviour of the participants, the sub-tasks that are absolutely crucial for using the feature.

As a general guideline, the moderators should not guide or assist the participants in any way – they are there only to take notes and ensure the participants attempt each task and fill out the survey.

After each task, each participant answers a question about the ease of the task and how confident he is that he successfully completed the task. These statements are graded on a 5 or 7-point scale.

Task instruction	Ease of task / Confidence																																				
Please collapse the video info widget	<table border="1"> <tr> <td data-bbox="762 584 887 611">Very difficult</td> <td colspan="7"></td> <td data-bbox="1257 584 1350 611">Very easy</td> </tr> <tr> <td></td> <td data-bbox="804 633 845 674">1</td> <td data-bbox="884 633 925 674">2</td> <td data-bbox="963 633 1005 674">3</td> <td data-bbox="1043 633 1085 674">4</td> <td data-bbox="1123 633 1165 674">5</td> <td data-bbox="1203 633 1244 674">6</td> <td data-bbox="1283 633 1324 674">7</td> <td></td> </tr> <tr> <td data-bbox="762 723 871 772">Not confident at all</td> <td colspan="7"></td> <td data-bbox="1257 723 1334 772">Totally confident</td> </tr> <tr> <td></td> <td data-bbox="804 779 845 819">1</td> <td data-bbox="884 779 925 819">2</td> <td data-bbox="963 779 1005 819">3</td> <td data-bbox="1043 779 1085 819">4</td> <td data-bbox="1123 779 1165 819">5</td> <td data-bbox="1203 779 1244 819">6</td> <td data-bbox="1283 779 1324 819">7</td> <td></td> </tr> </table>	Very difficult								Very easy		1	2	3	4	5	6	7		Not confident at all								Totally confident		1	2	3	4	5	6	7	
Very difficult								Very easy																													
	1	2	3	4	5	6	7																														
Not confident at all								Totally confident																													
	1	2	3	4	5	6	7																														
How confident are you that you successfully completed the task?																																					

The moderator should note whether the participant made the correct first click and whether the participant successfully completed the task.

After all tasks are completed, the participants are then asked to complete the SUS including the Usability Adjective Scale. This is a survey with 11 statements with an associated 7-point scale.

Graphical Editor User Trials Scenarios

WIDGET EDITOR SCENARIOS

1. Select the Test Wikipedia widget.
2. Change the background colour of the header in the Test Wikipedia widget's style. Save the change.
3. Choose a style that is not the default one and make it default.
4. Delete a style that is not the default one.
5. Create a new style.
6. Save a new style under a new name.
7. Export a CSS style to your computer.
8. Import a CSS style from your computer.
9. Load an HTML template from the HTML gallery and save it as the Wikipedia HTML.
10. Export an HTML template to your computer.
11. Import an HTML template from your computer.