EUROPEAN COMMISSION
Research Executive Agency (REA)
Inclusive, Innovative and Reflective Societies

Project acronym:	SIMPATICO
Project full title:	SIMplifying the interaction with Public Administration Through Information technology for Citizens and cOmpanies
Call identifier:	EURO-6-2015
Type of action:	RIA
Start date:	1 March 2016
End date:	28 February 2019
Grant agreement no:	692819

D5.3 – SIMPATICO Interoperability Framework and Use-Case Environments v1

WP5	Integration and Environment Setup
Due Date:	30/04/2017
Submission Date:	16/05/2017
Responsible Partner:	Engineering – Ingegneria Informatica SpA (ENG)
Version:	1.0
Status:	Final
Author(s):	Antonio Filograna (ENG), Vincenzo Savarino (ENG), Raman Kazhamiakin (FBK), Michele Trainotti (FBK), Raúl Santos de la Cámara (HIB), Unai Lopez (DEUSTO), Neetu Agrawal (SPARTA)
Reviewer(s):	Giuseppe Di Modica (BENG), Gabriele Zacco (FBK)
Deliverable Type:	OTH: Other
Dissemination Level:	PU: Public

Version History

Version	Date	Author	Partner	Description
0.1	01/03/2017	A. Filograna	ENG	First version of ToC.
0.2	24/03/2017	A. Filograna	ENG	Contribution to Section 2.
0.3	28/03/2017	R. Kazhamiakin A. Filograna	FBK ENG	Contribution to Section 3.3.
0.4	31/03/2017	U. Lopez V. Savarino	DEUSTO ENG	Contribution to Section 3.2.
0.5	05/05/2017	G. Di Modica G. Zacco	BENG FBK	First round of Internal review.
0.6	08/05/2017	All	All	Contribution to Section 4.
0.7	09/05/2017	A. Filograna	ENG	Introduction and Conclusion added.
0.8	10/05/2017	G. Di Modica G. Zacco	BENG FBK	Second round of Internal review.
0.9	11/05/2017	A. Filograna	ENG	Final version.
1.0	16/05/2017	M. Pistore	FBK	Final quality check.

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Simpatico

Table of contents

List of figures

List of tables

Glossary

EIF	European Interoperability Framework
CDV	Citizen Data Vault
UP	User Profile
TAE	Text Adaptation Engine
WAE	Workflow Adaptation Engine
IFE	Interactive Front-End
SF	Session Feedback
DA	Data Analysis
EE	Enrichment Engine
QAE	Question Answering Engine
CPD	Collaborative Procedure Design
CKB	Collective Knowledge Base
GE	Gamification Engine
DB	Dashboard
IL	Integration Layer
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
PA	Public Administration
SME	Small and Medium Enterprise
UI	User Interface
HTML	HyperText Markup Language
GUI	Graphical User Interface

RDF	Resource Description Framework
DB	Data Base
DOM	Document Object Model
XML	EXtensible Markup Language
AJAX	Asynchronous JavaScript and XML
JSON	JavaScript Object Notation
PDS	Personal Data Store
AAC	Authentication and Authorization Component
SSO	Single Sign On

Executive summary

This document is the deliverable "**D5.3 – SIMPATICO interoperability framework and use-case environments v1**" of the European project "SIMPATICO - SIMplifying the interaction with Public Administration Through Information technology for Citizens and cOmpanies" (hereinafter also referred to as "**SIMPATICO**", project reference: 692819).

SIMPATICO addresses a strategic challenge towards the innovation and modernization of the public sector: the need to offer a more efficient and more effective experience to companies and citizens in their daily interaction with Public Administration (PA) by providing a personalized delivery of e-services based on advanced cognitive system technologies and by promoting an active engagement of people for the continuous improvement of the interaction with these services. To this purpose, the project implements the **SIMPATICO Platform**, a software platform that collects and integrates the advanced techniques developed in the SIMPATICO project and enacts them on top of the existing PA systems for on-line service delivery.

The aim of this document is to define the **SIMPATICO interoperability framework**, starting from the European Interoperability Framework (EIF). In particular, SIMPATICO had taken into account two layer of the EIF: the semantic and technological interoperability. These two layers play an important role to ensure the possibility to integrate SIMPATICO with the legacy systems of the PAs. The first layer has been met through the adoption of a thesaurus in the Citizen Data Vault in order to make **semantically interoperable** the information requested by the e-service provided by the Public Administration and the Personal Data of the citizen in the three use-cases. Therefore, the semantic interoperability has been used also in the interaction between Interactive Front-End and Citizenpedia components to associate a "semantic meaning" to the terms in a document or form (coming from e-services), in order to link them correctly to the definitions in Question and Answer Engine and Citizenpedia. The **technical interoperability** is ensured by the adoption of Open APIs and the unique point of authentication for the whole SIMPATICO platform, sharing in all three Use-cases. In order to test all the APIs provided by the project components, a unique SwaggerUI service has been created where each component API is documented.

This deliverable also covers the **deployment** and the **integration activities** required to instantiate the platform in the SIMPATICO use-cases. This document describes the actions that each use-cases task force has undertaken for the deployment, along with the component owner involved in the process, thus providing also a reference for future deployments by other PAs.

1 Introduction of SIMPATICO

This deliverable presents the outcomes of SIMPATICO project task T5.2 "*SIMPATICO interoperability framework*" and T5.3 " *Use-case deployment and connection with legacy systems* " in the scope of WP5 "*Integration and environment setup*". During the first 14 months of project execution, this task has worked to the adaptation of European Interoperability Framework for SIMPATICO project and all the activities carried out to deploy, set up and integrate the SIMPATICO platform in order to be ready for the first phase of use-cases execution.
To better understand the aim and scope of this document, in this introductory section we provide an overview of the SIMPATICO project (Section 1.1) and a description of the structure of the rest of this deliverable (Section 1.2).

1.1 SIMPATICO project

SIMPATICO's goal is **to improve the experience of citizens and companies in their daily interactions with the public administration** by providing a personalized delivery of **e-services** based on advanced **cognitive system technologies** and by promoting an active engagement of people for the continuous improvement of the interaction with these services. The SIMPATICO approach is realized through a platform that can be deployed on top of an existing PA system and allows for **a personalized service delivery** without having to change or replace its internal systems: a process often too expensive for a public administration, especially considering the cuts in resources imposed by the current economic situation.
The goal of SIMPATICO is accomplished through a solution based on the **interplay of language processing, machine learning and the wisdom of the crowd** (represented by citizens, business organizations and civil servants) **to change for the better the way citizens interact with the PA. SIMPATICO adapts the interaction process** to the characteristics of each user; **simplifies** text and documents to make them understandable; **enables feedback for the users** on problems and difficulties in the interaction; **engages civil servants, citizens and professionals** so as to make use of their knowledge and integrate it in the system (Figure 1).

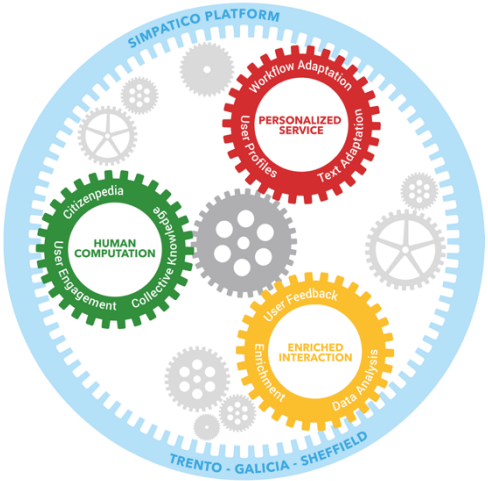

Figure 1: SIMPATICO concept as a glance

The project aims can be broken down into the following **smaller research objectives (ROs)**.

RO1. Adapt the interaction process with respect to the profile of each citizen and company (PA service consumer), in order **to make it clear, understandable and easy to follow**.

- A **text adaptation** framework**,** based on a **rich text information layer** and on machine learning algorithms capable of **inducing general text adaptation operations** from **few examples, and of customizing these adaptations to the user profiles.**
- **A workflow adaptation engine** that takes user characteristics and tailor the interaction according to the user's profile and needs.
- A feedback and annotation mechanism that **gives users the possibility to visualize, rate, comment, annotate, document the interaction process** (e.g., underlying the most difficult steps), so as to provide valuable feedback to the PA, further refine the adaptation process and enrich the interaction.

RO2. Exploit the wisdom of the crowd to enhance the entire e-service interaction process.

- An **advanced web-based social question answering engine (Citizenpedia)** where citizens, companies and civil servants **discuss and suggest potential solutions and interpretation for the most problematic procedures and concepts.**
- A **collective knowledge** database on e-services used to simplify these services and improve their understanding.
- An **award mechanism** that **engages users and incentivizes them to collaborate** by giving them **reputation** (a valuable asset for professionals and organizations) and **privileges** (for the government of Citizenpedia – a new public domain resource) according to their contributions.

RO3. Deliver the SIMPATICO Platform, an open software system that can interoperate with PA legacy systems.

- A platform that **combines consolidated e-government methodologies with innovative cognitive technologies** (language processing, machine learning) at different level of maturity, enabling their experimentation in more or less controlled operational settings.
- An interoperability platform that enables an **agile integration of SIMPATICO's solution with** PA legacy systems and that allows the exploitation of data and services from these systems with the SIMPATICO adaptation and personalization engines.

RO4. Evaluate and assess the impact of the SIMPATICO solution

- Customise, deploy, operate and evaluate the SIMPATICO solution on **three use-cases in two EU cities** – Trento (IT) and Sheffield (UK) – **and one EU region** – Galicia (ES).
- **Assess the impact** of the proposed solution in terms of **increase in competitiveness, efficiency of interaction and quality of experience.**

This deliverable focuses in particular on RO3 and on RO4. For what concerns RO3, the document covers the aspects related to the interoperability framework. For what concerns RO4, the document covers the activities performed for the integration and set up of the SIMPATICO platform in the three use-cases.

1.2 Structure of the deliverable

The remainder of the deliverable is organized as follows:

Section 2 describes the key concepts of European Interoperability Framework.

Section 3 describes the tailoring and accordingly the adoption of European Interoperability Framework by SIMPATICO project.

Section 4 describes the activities performed for the integration and set up of the SIMPATICO platform in the three use-cases.

Section 5 defines the SIMPATICO platform Deployment Plan and the activities carried out in the three use-cases.

2 European Interoperability Framework: key concepts

In 2013 the European Digital Agenda required all member States to adopt the European Interoperability Framework (EIF). This framework aims at helping the Public Administrations (PA) of the States across the Europe to connect information systems. The main goal of the EIF is to provide a set of recommendations and best practices to facilitate the interoperability activities. The framework would “be based on open standards and encourage the use of open source software” [1].

The standard definition of interoperability is "the ability of software and hardware on multiple machines from multiple vendors to communicate" [2].
Therefore, the meaning of the interoperability word is not to be intended in a technological sense; rather, it mainly refer to the ability of people, ICT systems and business process to work together and interact each other in order to provide services to citizens and businesses in an integrated way".
An **interoperability framework** can be defined as a **set of standards and guidelines** that describes the way in which organisations have agreed, or should agree, to interact with each other" [1].

The **European Interoperability Framework** grounds on the latter definition and provides a set of recommendations and best practices to guide the development of eGovernment e-services enabling all the actors (citizen, enterprises, PAs) to interact to each other and across borders. The EIF has to meet several requirements like:

- **Accessibility**: the eGovernament services have to be accessible by everyone without discrimination. The use for disabled people has to be taken into account, as well as a multichannel approach to give the customer the possibility to choose the preferred interaction mean (e.g. web, kiosk, mobile, etc.) following the eInclusion approach.
- **Multilingualism**: since a wide variety of languages are used, the back-end of the e-services has to be independent from the used language. If it is not possible, a sort of translation mechanism has to be put in place. The multilingualism must not be an obstacle to the provision of e-services.
- **Security**: the security has to be guaranteed performing a suitable risk assessment. From the user point of view, all security mechanisms have to be provided as transparently as possible.
- **Privacy**: each user has to know how their personal data will be used and for which purpose. All systems have to be compliant with the current EU data protection regulation.
- **Subsidiarity**: the aim of EIF is to not interfere with the internal work of EU and local public administrations. Moreover, the single public administrations have to guarantee the interoperability at European level.
- **Use of Open Standards**: the use of open standards is favoured. The standard has to be managed by an open community and its specification has to be freely available.
- **Assess the benefits of Open Source Software**: accordingly, the use of Open Source Software has to be taken into account. In this way, the benefit of this usage will be assessed.
- **Use of Multilateral Solutions**: the partners involved in the cooperation should adopt the same set of agreement for the interoperability solution. In doing so, they can avoid to waste time in one-to-one communication, thus getting the system to reduce the costs in a more efficient way.

Starting from the aforementioned principles, the EIF can be divided in three sub-categories: Organizational Interoperability, Semantic Interoperability and Technical Interoperability.

The next three sections aim at giving a brief description of each of these three sub-categories. Further information can be found in [1].

2.1 Organizational Interoperability

The Public Administrations provide citizens and enterprises with eGovernment services to help them running their life and business, respectively. The service consumers have to use these services in a transparent way without taking care about the PAs administrative procedures running in the backend.

The organizational interoperability focuses on business goals, modelling business processes and trying to make PAs of different Member States interoperable from the administrative procedure point of view. Considering the subsidiarity principle, it is necessary to know the entry and exit points of the procedures provided by different Member State administrations in order to deliver the service requested by a citizen or enterprise. In doing so, the organizational interoperability can take place by implementing and documenting the *Business Interoperability Interfaces (BII)* that allow two different Member States PAs to communicate without changing their own internal business processes.

The communication, the agreement and the study of the services and actors involved in the process must be analysed and documented by all the different administrations involved in the organizational pathway. In fact, in providing these eGovernment services, a service level agreement on quality, time delivery, security and data protection must be guaranteed to the service consumer.

The SIMPATICO project does not deal with organizational interoperability because providing cross-border services is out of the scope of the project. This is the reason why the customization of the Organizational Interoperability for the SIMPATICO project will not be addressed in Section 3.

2.2 Semantic Interoperability

The semantic interoperability offers the possibility to elaborate information by external sources while keeping the real meaning of the information unchanged during the elaboration process. The meaning is preserved in order that all the actors involved in the process (people, institutions, applications) can understand it.

In other words, the semantic interoperability involves, within the specific sector, the definition of common data schema and protocols. Basically, the EIF specifies the actions to be undertaken by Member States to guarantee the semantic interoperability in the cooperation and interchange among organizations.

This kind of actions leverages on:

- both general and sector-specific tools, to support the information sharing;
- basic and clear principles for the management of information kept by the Governments;
- a common set of data schema and protocols, taxonomies for specific sectors; the national framework should take into account the cross-border interoperability when defining these taxonomies, using standard data model that can be understood and used also in another countries;
- protocols for sharing and reusing information through public and private sectors;
- processes for the management of organizations' information life cycle.

As mentioned in [1], the main requirement to perform the semantic interoperability is to have a common language among all Member States. This language has to describe in a meaningful way the

structure of the involved data. To this end, it is necessary to select a mark-up language like the XML. In particular, the XML-Schema and its related artefacts (e.g. ontology, metadata) enables the integration of services developed with different vocabularies and different vision on the data.

2.3 Technical Interoperability

The technical interoperability deals with the communication aspects among different computer systems and services. Open interfaces, data integration, middleware, accessibility and security are the main key aspects.

Following a survey on Information Needs of Different EU Stakeholders [3], the knowledge of technical interoperability is wider than that of organizational and semantic among the European stakeholders in interoperability of Member States. This means most of them consider this aspect not a priority. That is why there are many government services available online, from the informative ones to those that allow multiple transactions. Moreover, the technical interoperability is granted on the last ones, i.e., services that can be used online and that connect different Public Administrations from different Member States, being fully automated.

The technical integration can happen both at front-end and back-end level. The **front-end integration** takes into account aspects such as [1]:

- Data presentation and exchange
- Accessibility - Interface design principles
- Multi-channel access
- Character sets
- Collective authoring
- File type and document formats
- File compression

The **back-end integration** addresses the following aspects:

- Data integration and middleware
- XML-based standards
- EDI-based standards
- Web Services
- Distributed Application Architecture
- Interconnection services
- File and message transfer protocols
- Message transport and security
- Message store services
- Mailbox access
- Directory and domain name services
- Network services

Also, the following **security aspects** need to be considered:

- Security services
- General security services - PKI
- Web service security
- Firewalls

- Protection against viruses, worms, Trojan horses and e-mail bombs

Therefore, in order to achieve the technical interoperability, it is at least necessary the use of universally agreed open standards and specifications (e.g. the family of XML standards).

3 SIMPATICO Interoperability Framework

Starting from European Interoperability Framework, briefly described in Section 2, this Chapter describes the Interoperability Framework tailored for SIMPATICO project.
The purpose of SIMPATICO project does not foresee a collaboration among Public Administrations in different countries. This is the reason why the Organizational Interoperability will not be tailored for SIMPATICO project as stated in the Section 3.1 (below).
On the contrary, a description of what actions will be undertaken to follow the guidelines and recommendations given by EIF in terms of the Semantic and Technical Interoperability will be provided in Sections 3.2 and 3.3.

3.1 Organizational Interoperability

The SIMPATICO project does not deal with organizational interoperability since providing cross-border services is out of the scope of the project.

3.2 Semantic Interoperability

The aim of SIMPATICO project is to ease the usage of the public e-services, helping the citizens or enterprises to interact with the public administration. One of the goal to attain is the simplification of the public e-services. In doing so, two main actions will be undertaken in order to make e-services semantically interoperable with the components of the SIMPATICO platform, thus guaranteeing the simplification. In particular, the Citizen Data Vault (CDV) deals with the citizen and/or enterprise personal data; SIMPATICO makes understandable every field of the e-service's online form by automatically filling the form with the citizen personal data directly loaded from the CDV.

The second action works towards the interoperability between the Interactive Front-End and the Citizenpedia. The goal is to associate a "semantic meaning" to the terms in a document or form (coming from e-services), in order to correctly link them with the definitions in Q&A and Citizenpedia. Similarly, links within Citizenpedia also require a semantic meaning. In the future, many proposals for standard taxonomies for eGov services and procedures will be analysed since semantic relations are necessary to integrate them (and use them as sources for Citizenpedia).

3.2.1 Citizen Data Vault and Taxonomy

Within the SIMPATICO Platform, the Citizen Data Vault (CDV) represents the component that will take care of the storage of personal data of the user, and of the exchange of these data among the user, the legacy systems and e-services of the PA, and other components of the SIMPATICO Platform. Primary consumer of such data will be the Interactive Front End (IFE), through which the CDV repository is continuously updated during each interaction of users with the PA e-services and is further invoked to automatically pre-fill e-services forms. In this way the citizen/company will give the PA the same information only once, as the information will be stored in the vault and used in all the following interactions.

The CDV interaction flow has been designed to ensure a semantic interoperability that enables that all data collected during the interactions can be reused among all e-services, without any intervention on them. According to this interaction flow, each PA e-service willing to interact with the CDV has to:

- Be registered on the CDV(*Service Registration*);
- Be assigned an Account (*Service Linking*);
- Get a "consent" to access data (*Data Consent*).

It is in the first phase (Service Registration) that semantic interoperability is ensured. This phase produces a description of the service and in particular the mapping of the service dataset with categories and data fields from a taxonomy of Personal Data.

In order to achieve this goal, a set of Personal Data fields has been defined, belonging to the following categories:

- *Identity*
- *Relationship*
- *Activity*
- *Health*
- *Financial*
- *Government*
- *Content*
- *Context*
- *Communications.*

A service is first registered providing a set of metadata describing the service itself. Once registered, a description of the service data is provided on which there is a "mapping" with the concepts of the Personal Data Taxonomy. By the term “data” we refer to the data that it consumes by interacting with CDV, for example a set of fields in a PA web form (Figure 2).

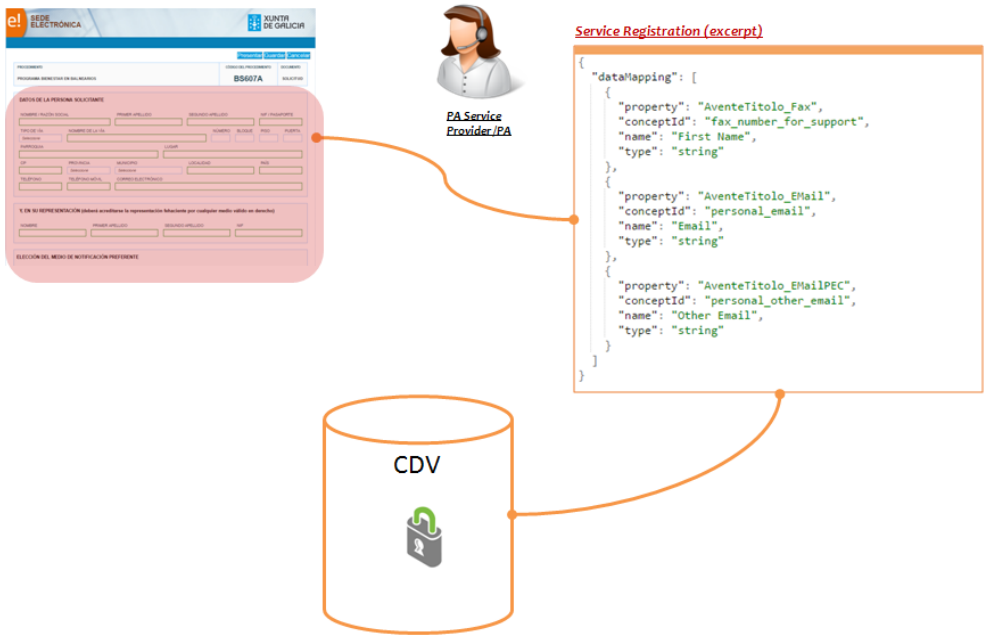

Figure 2: Service Registration phase where the service data is mapped with a set of concepts belonging to a Personal Data Taxonomy

The Service Manager is the components of CDV in charge of storing the service description (Figure 3) and in particular of providing the service data concepts mapping during the interaction of the CDV with the e-services.

Figure 3: Service Manager stores the service description and it provides concepts mapping during the interaction of CDV with e-services.

In fact, each information provided during the interaction of users with PA e-services will be saved as values set of a specific concept in order to assure that all data collected or retrieved belong to a common data schema (Figure 4).

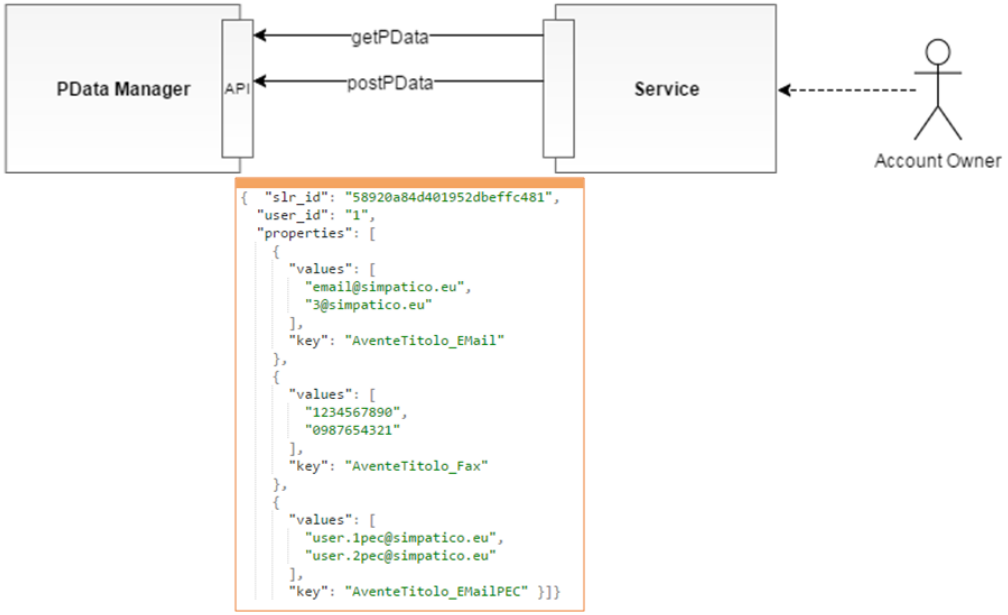

Figure 4: Data Store/Retrieve starting from Data Mapping in which each value is mapped to a specific concept

This enables, for example, that a personal data set stored during the interaction of a user with a service "A" can be retrieved and processed during the interaction of the same user with a service "B" (Figure 5).

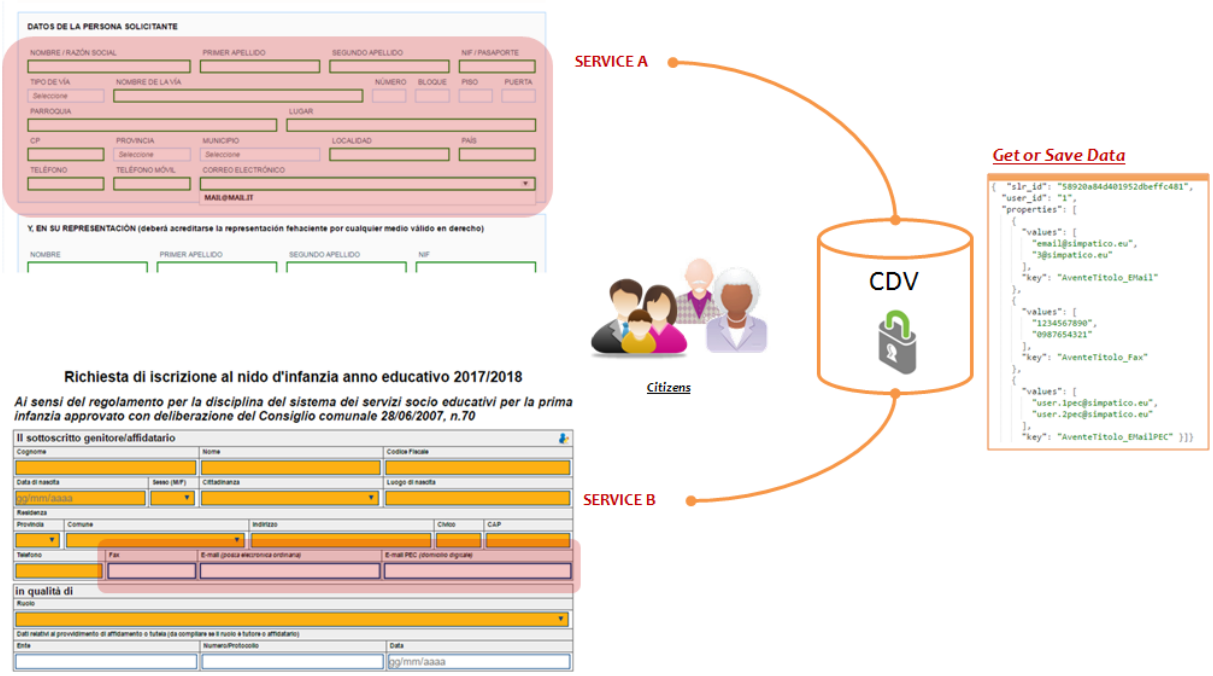

Figure 5: Semantic interoperability permits that data values stored during the interaction with service A can be retrieved during the interaction with Service B

3.2.2 Interaction Front-End and Citizenpedia

This section details the interactions between the Interactive Front-end and the Citizenpedia. Given that these components represent the main interaction point of the users with the Simpatico platform, their seamless integration must be carefully deployed.

One of the key aspects of Semantic Interoperability is to ensure that different software pieces "speak the same language", i.e., in this context that the same data/concept has the same meaning in both components (e.g. that a PA-related-question means the same in the IFE and in the Citizenpedia). In this section we describe how this "common meaning" has been achieved, and how the communication between IFE and Citizenpedia is implemented.

The first step to ensure a common semantic has been to specify a vocabulary that defines the terms used in these interactions. These terms are:

- *Procedure*: It is a process defined by the public administration which contains several steps represented by a Collaborative Procedure Designer (CPD) diagram.
- *E-Service*: Is an electronic service used to complete one step to achieve one procedure. It is represented by a CPD diagram element (e.g. a node).
- *Diagram element:* It is a graphical element of a Diagram representing a specific component of a procedure. For instance, it may represent: a) an entire e-service, a part of it or a subcomponent inside it (e.g. a form); b) a face-to-face interaction between the citizen and a civil servant; c) a decisional step which can divert the procedure flow.
- *Paragraph*: Is a text describing a part of an e-service, law, or instructions inside them.

On top of these terms, several interactions between IFE and Citizenpedia were defined. In order to make them more understandable, the different actions that can happen between them have been

described in a diagram. In addition, the interactions with the two modules of Citizenpedia were separated: QAE and CPD. The Figure 6 represents these interactions:

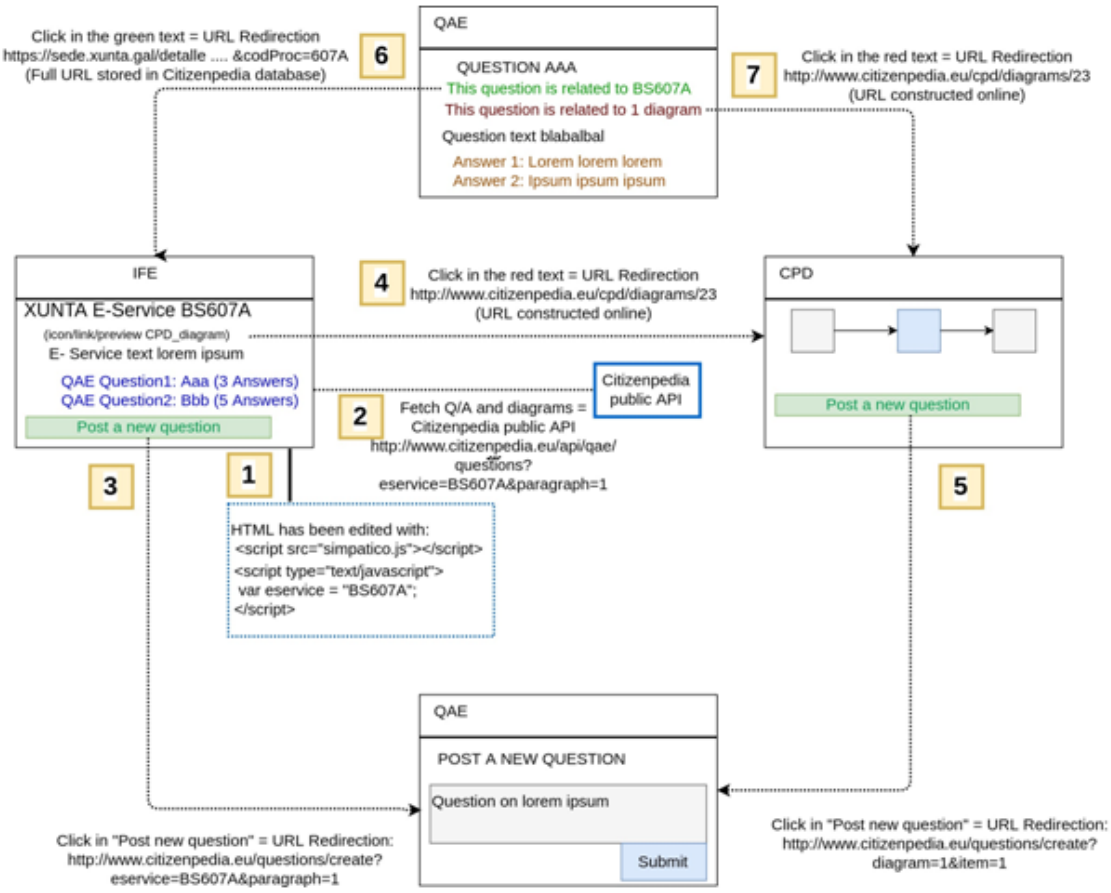

Figure 6: Interaction between IFE and Citizenpedia

For the sake of clarity, the interactions are defined in the following list. Each numbered item of the list corresponds to an action of the figure:

1. The interaction with the IFE requires to include the text in the blue box within the HTML of the original e-service. This also will require the injection of the JavaScript library and the addition of a variable called “e-service”.

The “e-service” variable will be used to univocally identify the e-service in the Citizenpedia. The value of the variable, i.e. the code (e.g. BS607A), is decided by the IT administrator of the municipality. In the example, it matches the real code of the e-service

2. Using the “e-service” code, the IFE calls a method of the public API of the Citizenpedia in order to fetch the related questions, answers and the CPD diagram containing the diagram element representing the “e-service”.

3. When the user clicks on the green button, the user is redirected to a Citizenpedia's html form to create a new question. This new html form is reached through a URL created dynamically with the following form: CITIZENPEDIA_NEW_QUESTION_URL+PARAMETERS. The parameters include the "e-service" code and the related paragraph number, and they are used to add this information to the new question.

4. When the user clicks the button with the link to the related diagram, the user is redirected to the related diagram in the CPD. The URL of this form is constructed dynamically with the following form: CITIZENPEDIA_CPD_DIAGRAM_URL+DIAGRAM_CODE. The diagram code is fetched from the Citizenpedia database in the query of step 2.

5. In the CPD, when the user clicks the button to post a new question on a specific diagram element, he gets redirected to an html form in the QAE. Each diagram element has its own button to post a question. It is created in the same way as in step 3, but instead of an "e-service" code, a diagram code is used. The diagram element code is an alpha-numeric string.

6. In an existing question of the QAE, by clicking the button of the related e-service, opens the original e-service will open up. The URL of this e-service is stored in the Citizenpedia database.

7. Similarly to step 4, when the user clicks the button with the link to the related diagram element, s/he is redirected to the related diagram in the CPD.

For better understanding the interactions, we will get into the insights of the Citizenpedia database. This database follows the classic Entity-relationship model, whose model is depicted in the Figure 7.

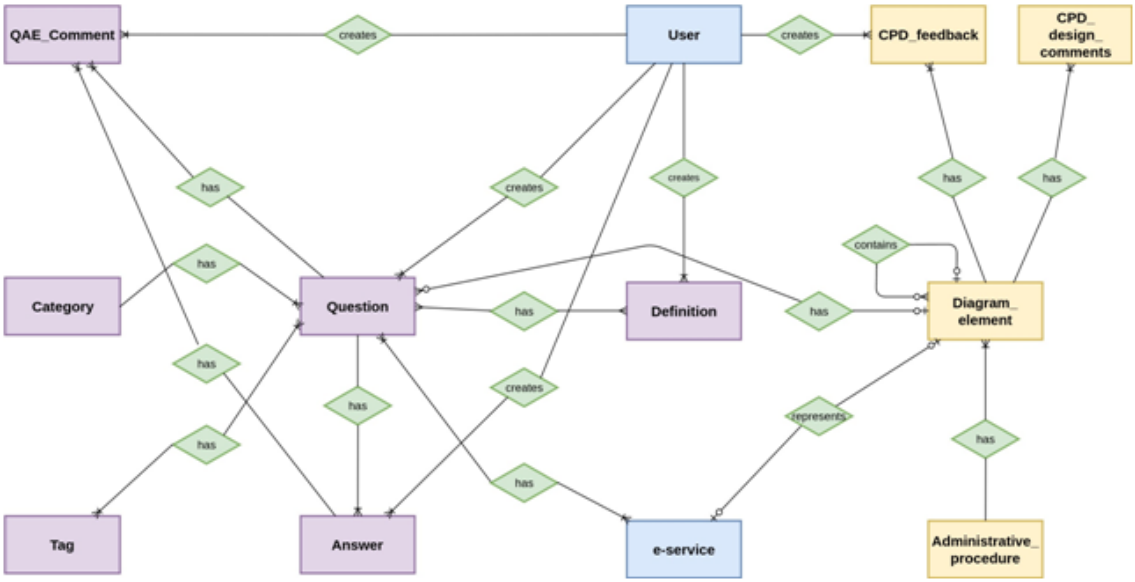

Figure 7: Entity-relationship model

In the Figure 7, the entities related to the QAE were represented in purple, the ones related to the CPD in yellow, and the shared ones that also serve for interoperability in blue. The attributes of the entities were omitted for sake of simplicity. In addition, it is important to highlight the following entity (Figure 8):

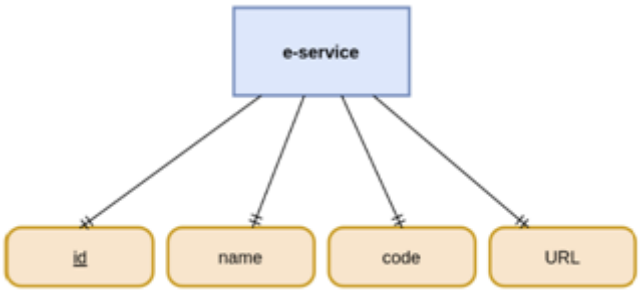

Figure 8: e-service entity

The "e-service" code variable is stored in the "code" field of the "e-service" table, and it enables to process the links between IFE and Citizenpedia.

For now, this way of operating allows the interoperability between these components. However, additional semantic techniques could be adopted to enhance their interoperability with 3rd party services. In particular, this is the translation of the model and taxonomy to RDF and XML resources standards, as defined by the Core Vocabularies Specification of the EU.

3.3 Technical Interoperability

The actions to follow the recommendations of EIF to make SIMPATICO technically interoperable have focused on the development and usage of Open APIs and the unique point of authentication for the whole SIMPATICO platform, shared in all of the three Use-cases.

3.3.1 Open APIs

To follow the recommendations provided by the EIF in terms of technical interoperability, all the SIMPATICO component APIs are provided as Open APIs.

Every APIs has been documented by using the Swagger notation[4]. A unique point of access to all the component APIs has been created as a SwaggeUI service, where it is possible to test the API and understand its use and scope. The SIMPATICO SwaggerUI service is reachable at the following link:

https://simpatico.eng.it/SwaggerUI/

Below, a brief component by component description of the APIs and their related input parameters is provided.

3.3.1.1 Citizen Data Vault

The APIs provided by CDV are divided in three main class: PData Manager, Account Manager and Service Manager.

PData Manager APIs

These APIs provide functionalities for managing the **Personal Data of an Account Owner**. In addition, they provide functionalities for storing and getting these Personal Data, starting from the properties required by a previously linked service.

Table 1: PData Manager APIs

Operation & method	Operation Description	Input	Input Description
/pData - DELETE	Deletes all the Account owner's Personal Data	accountId (string)	The identifier of the Account owner
/pData - GET	Gets all the Account owner's Personal Data in several formats.	accountId (string)	The identifier of the Account owner
/pData - POST	Stores one or more Account owner's Personal Data	accountId (string)	The identifier of the Account owner
		request (json)	An array of objects containing the Personal Data to be stored
/pData - PUT	Updates one or more Account owner's Personal Data	accountId (string)	The identifier of the Account owner
/pData/Download - GET	Gets the dump file of all the Account and related Personal Data, in either CSV or JSON	accountId (string)	The identifier of the Account owner
/pData/{conceptId} - DELETE	Deletes a specific Personal Data of Account owner	accountId (string)	The identifier of the Account owner
		conceptId	The identifier of the semantic concept related to Personal Data
/pData/{conceptId} - GET	Gets a specific Account owner's Personal Data	accountId (string)	The identifier of the Account owner
		conceptId	The identifier of the semantic concept related to Personal Data
/pData/{conceptId} - PUT	Updates a specific Account owner's	accountId (string)	The identifier of the Account owner

	Personal Data	conceptId	The identifier of the semantic concept related to Personal Data
/postPData - POST	Stores one or more Account Owner's Personal Data for the linked Service	request (json)	An object containing the service properties to be stored as Account Owner's Personal Data, after resolving the mapping.
/getPData - POST	Gets one or more Account owner's Personal Data for the linked Service	request (json)	An object containing the service properties to be stored as Account Owner's Personal Data, after resolving the mapping.

Account Manager APIs

The list of following APIs provide functionalities for managing the **Account Owner**'s account. In addition, they provide functionalities for performing the linking process between a Service and the Account Owner's account. They are divided in sub-groups: Internal, External, Account Owner and Service.

The APIs in Table 2 are used for communications between CDV's **internal** components and its frontend.

Table 2: Account Manager APIs - Internal

Operation & method	Operation Description	Input	Input Description
/accounts - POST	Creates a new Account	request (json)	Object representing the Account to be created
/accounts/{accountId} - DELETE	Deletes an existing Account and all related Personal Data	accountId (string)	The unique identifier of the Account to be retrieved
/accounts/{accountId} - GET	Gets an existing Account, with	accountId (string)	The unique identifier of the Account to be

	optionally all the related Personal Data		retrieved
		withPData (boolean)	The flag (True/False) to retrieve the Personal Data related to the requested account
/accounts/{accountId} - PUT	Updates an existing Account	accountId (string)	The unique identifier of the Account to be retrieved
/accounts/{accountId}/download - GET	Download the dump file of an existing Account with all the related Personal Data, in JSON format	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/telephones - GET	Gets all the Telephones of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/telephones - POST	Creates a new Telephone for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Telephone to be created
/accounts/{accountId}/telephones/{telephoneId} - DELETE	Deletes an existing Telephone of an existing Account	accountId (string)	The unique identifier of the Account
		telephoneId	The unique identifier of the Telephone to be

			deleted
/accounts/{accountId}/telephones/{telephoneId} - GET	Gets a specific Telephone of an existing Account	accountId (string)	The unique identifier of the Account
		telephoneId	The unique identifier of the Telephone to be deleted
/accounts/{accountId}/telephones/{telephoneId} - PUT	Updates an existing Telephone of an existing Account	accountId (string)	The unique identifier of the Account
		telephoneId	The unique identifier of the Telephone to be deleted
/accounts/{accountId}/contacts - GET	Gets all the Contacts of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/contacts - POST	Creates a new Contact for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Contact to be created
/accounts/{accountId}/contacts/{contactId} - DELETE	Deletes an existing Contact of an existing Account	accountId (string)	The unique identifier of the Account
		contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/contacts/{contactId} - GET	Gets a specific Contact of an	accountId (string)	The unique identifier of the

	existing Account		Account
		contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/contacts/{contactId} - PUT	Updates an existing Contact of an existing Account	accountId (string)	The unique identifier of the Account
		contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/emails - GET	Gets all the Emails of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/emails - POST	Creates a new Email for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Email to be created
/accounts/{accountId}/emails/{emailId} - DELETE	Deletes an existing Email of an existing Account	accountId (string)	The unique identifier of the Account
		emailId	The unique identifier of the Email to be deleted
/accounts/{accountId}/emails/{emailId} - GET	Gets a specific Email of an existing	accountId (string)	The unique identifier of the Account

	Account		
		emailId	The unique identifier of the Email to be retrieved
/accounts/{accountId}/emails/{emailId} - PUT	Updates an existing Email of an existing Account	accountId (string)	The unique identifier of the Account
		emailId	The unique identifier of the Email to be updated
/accounts/{accountId}/particular - DELETE	Deletes the Particular of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/particular -GET	Gets the Particular of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/particular -PUT	Creates a new Particular for an existing Account, overwriting, if any, the existing one.	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Particular to be created
/accounts/{accountId}/serviceLinks - GET	Gets all the Service Link Records of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/serviceLinks - POST	Creates a new Service Link Record for an	accountId (string)	The unique identifier of the Account

	existing Account.		
		request (json)	Object representing the Service Link Record to be created
/accounts/{accountId}/services/{serviceId} /serviceLinks - GET	Gets a specific Service Link Record of an existing Account and specific Service	accountId (string)	The unique identifier of the Account
		serviceId (string)	The unique identifier of the specific Service linked with the Account
/accounts/{accountId}/serviceLinks/{slrId} - DELETE	Deletes a specific Service Link Record of an existing Account	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique identifier of the Service Link Record to be deleted
/accounts/{accountId}/serviceLinks/{slrId} - GET	Gets a specific Service Link Record of an existing Account	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique identifier of the Service Link Record to be retrieved
/accounts/{accountId}/serviceLinks/{slrId} - PUT	Disables a specific Service Link Record of an existing Account.	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique

			identifier of the Service Link Record to be updated

The APIs in Table 3 are used for CDV interaction with **external** entities.

Table 3: Account Manager APIs - External

Operation & method	Operation Description	Input	Input Description
/accounts - POST	Creates a new Account	request (json)	Object representing the Account to be created
/accounts/{accountId} - GET	Gets an existing Account, with optionally all the related Personal Data	accountId (string)	The unique identifier of the Account to be retrieved
		withPData (boolean)	The flag (True/False) to retrieve the Personal Data related to the requested account
/users/{surrogateId}/services/ {serviceId}/serviceLink - DELETE	Deletes a specific Service Link Record of an existing Account and specific Service, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service
		serviceId	The unique identifier of the specific Service linked with the Account
/users/{surrogateId}/services/ {serviceId}/serviceLink - GET	Gets a specific Service Link Record of an existing Account and specific Service, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service
		serviceId	The unique identifier of the specific Service linked with the

			Account
/users/{surrogateId}/serviceLink - GET	Gets a Service Link Record of an existing Account, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service

The APIs in Table 4 are used by the CDV **Account Owner**, enabling them to perform management operations on its own Account and related Service Links.

Table 4: Account Manager APIs - Account Owner

Operation & method	Operation Description	Input	Input Description
/accounts - POST	Creates a new Account	request (json)	Object representing the Account to be created
/accounts/{accountId} - DELETE	Deletes an existing Account and all related Personal Data	accountId (string)	The unique identifier of the Account to be retrieved
/accounts/{accountId} - GET	Gets an existing Account, with optionally all the related Personal Data	accountId (string)	The unique identifier of the Account to be retrieved
		withPData (boolean)	The flag (True/False) to retrieve the Personal Data related to the requested account
/accounts/{accountId} - PUT	Updates an	accountId	The unique

	existing Account	(string)	identifier of the Account to be retrieved
/accounts/{accountId}/download - GET	Download the dump file of an existing Account with all the related Personal Data, in JSON format	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/telephones - GET	Gets all the Telephones of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/telephones - POST	Creates a new Telephone for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Telephone to be created
/accounts/{accountId}/telephones/{telephoneId} - DELETE	Deletes an existing Telephone of an existing Account	accountId (string)	The unique identifier of the Account
		telephoneId	The unique identifier of the Telephone to be deleted
/accounts/{accountId}/telephones/{telephoneId} - GET	Gets a specific Telephone of an existing Account	accountId (string)	The unique identifier of the Account
		telephoneId	The unique identifier of the Telephone to be deleted
/accounts/{accountId}/telephones/{telephoneId} - PUT	Updates an existing	accountId (string)	The unique identifier of the

	Telephone of an existing Account		Account
		telephoneId	The unique identifier of the Telephone to be deleted
/accounts/{accountId}/contacts - GET	Gets all the Contacts of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/contacts - POST	Creates a new Contact for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Contact to be created
/accounts/{accountId}/contacts/{contactId} - DELETE	Deletes an existing Contact of an existing Account	accountId (string)	The unique identifier of the Account
		contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/contacts/{contactId} - GET	Gets a specific Contact of an existing Account	accountId (string)	The unique identifier of the Account
		contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/contacts/{contactId} - PUT	Updates an existing Contact of an existing	accountId (string)	The unique identifier of the Account

	Account	contactId	The unique identifier of the Contact to be deleted
/accounts/{accountId}/emails - GET	Gets all the Emails of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/emails - POST	Creates a new Email for an existing Account	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Email to be created
/accounts/{accountId}/emails/{emailId} - DELETE	Deletes an existing Email of an existing Account	accountId (string)	The unique identifier of the Account
		emailId	The unique identifier of the Email to be deleted
/accounts/{accountId}/emails/{emailId} - GET	Gets a specific Email of an existing Account	accountId (string)	The unique identifier of the Account
		emailId	The unique identifier of the Email to be retrieved
/accounts/{accountId}/emails/{emailId} - PUT	Updates an existing Email of an existing Account	accountId (string)	The unique identifier of the Account
		emailId	The unique

			identifier of the Email to be updated
/accounts/{accountId}/particular - DELETE	Deletes the Particular of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/particular -GET	Gets the Particular of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/particular -PUT	Creates a new Particular for an existing Account, overwriting, if any, the existing one.	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Particular to be created
/accounts/{accountId}/serviceLinks - GET	Gets all the Service Link Records of an existing Account	accountId (string)	The unique identifier of the Account
/accounts/{accountId}/serviceLinks - POST	Creates a new Service Link Record for an existing Account.	accountId (string)	The unique identifier of the Account
		request (json)	Object representing the Service Link Record to be created
/accounts/{accountId}/services/{serviceId} /serviceLinks - GET	Gets a specific Service Link Record of an existing Account and specific	accountId (string)	The unique identifier of the Account
		serviceId	The unique

	Service	(string)	identifier of the specific Service linked with the Account
/accounts/{accountId}/serviceLinks/{slrId} - DELETE	Deletes a specific Service Link Record of an existing Account	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique identifier of the Service Link Record to be deleted
/accounts/{accountId}/serviceLinks/{slrId} - GET	Gets a specific Service Link Record of an existing Account	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique identifier of the Service Link Record to be retrieved
/accounts/{accountId}/serviceLinks/{slrId} - PUT	Disables a specific Service Link Record of an existing Account.	accountId (string)	The unique identifier of the Account
		slrId (string)	The unique identifier of the Service Link Record to be updated

The APIs in Table 5 are used by the **Service** or the Service Provider to manage and retrieve information about the Service Links and Accounts in which its users are involved.

Table 5: Account Manager APIs - Service

Operation & method	Operation Description	Input	Input Description

/accounts - POST	Creates a new Account	request (json)	Object representing the Account to be created
/accounts/{accountId} - GET	Gets an existing Account, with optionally all the related Personal Data	accountId (string)	The unique identifier of the Account to be retrieved
		withPData (boolean)	The flag (True/False) to retrieve the Personal Data related to the requested account
/users/{surrogateId}/services/ {serviceId}/serviceLink - DELETE	Deletes a specific Service Link Record of an existing Account and specific Service, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service
		serviceId	The unique identifier of the specific Service linked with the Account
/users/{surrogateId}/services/ {serviceId}/serviceLink - GET	Gets a specific Service Link Record of an existing Account and specific Service, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service
		serviceId	The unique identifier of the specific Service linked with the Account
/users/{surrogateId}/serviceLink - GET	Gets a Service Link Record of an existing Account, starting from the existing user Id (surrogateId) in the service.	surrogateId	The unique identifier of the User at the Service

The Service Manager APIs provide functionalities for managing the registration and mapping of e-services.

Table 6: Service Manager APIs

Operation & method	Operation Description	Input	Input Description
/services - GET	Get the list of registered services		
/services - POST	Stores a new Service Registration	request (json)	Object representing the Service Description
/services/{serviceId} - DELETE	Deletes an existing Service Description	serviceId	The unique identifier of the Service to be deleted
/services/{serviceId} - GET	Gets an existing Service Description	serviceId	The unique identifier of the Service to be retrieved
/services/{serviceId} - PUT	Updates an existing Service Description	serviceId	The unique identifier of the Service to be updated
		request (json)	Object representing the Service Description
/services/{serviceId}/servicedatamapping - GET	Provides the data mapping with personal data taxonomy.	serviceId	The unique identifier of the Service to be retrieved

3.3.1.2 Citizenpedia

Operation & method	Operation Description	Input	Input Description

/stats/questions/{eservice} - GET	This method counts the number of questions related to the properties provided as parameters	eservice (string)	The code of e-service
/stats/questions/{eservice}/{paragraph} - GET	This method counts the number of questions related to the properties provided as parameters	eservice (string)	The code of e-service
		paragraph (string)	Paragraph of the question
/qae/questions/{eservice} - GET	This method gets all the questions related to the properties provided as parameters	eservice (string)	The code of e-service
/qae/questions/{eservice}/{paragraph} - GET	This method gets all the questions related to the properties provided as parameters	eservice (string)	The code of e-service
		paragraph (string)	Paragraph of the question
/qae/term - GET	This method gets the definition of the provided term	term (string)	term

3.3.1.3 Interaction LOG

Operation & method	Operation Description	Input	Input Description
/find - GET	Find documents by words	words (string)	Comma separated words. Spaces are allowed
		sortasc (string)	Created time ascending sort

		sortdesc (string)	Created time descending sort
		limit (double)	Limit the number of results
/insert - POST	Insert one document. If in JSON there are '_id' key, the document will insert with these id.	query (json)	A valid JSON to insert.
/remove - DELETE	Delete one document. JSON with 'id' key is required.	query (json)	A valid JSON with 'id' key
/update - PUT	Update document's fields. JSON with 'id' and 'content' keys is required.	query (json)	A valid JSON with 'id' and 'content' keys

3.3.1.4 Collaborative Procedure Design

Operation & method	Operation Description	Input	Input Description
/stats/diagram/{diagramId}/eServiceCount - GET	Fetches the number of all the *DiagramElements* that are associated to an e-service and belong to the *Diagram* identified by {diagramId}.	diagramId (string)	ID of diagram
/diagram/eService/{eServiceId}/summary - GET	Retrieves the *DiagramSummary* for the given e-service {eServiceId}.	eServiceId (string)	id of e-service
/diagram/summary/list - GET	Retrieves the list of all *DiagramSummary*	-	-

3.3.1.5 Text Adaptation Engine

Operation &	Operation Description	Input	Input Description

method			
/tae/simp - GET	Obtain annotations and simplifications	word (string)	word to be simplified, if any
		position (integer)	word position in the context, in case of word simplification
		lang (string)	language, as 2-letter ISO code. If not specified, derived by the tool
		text (string)	sentence or text to be simplified in case of syntactic simplification. Word context in case of word simplification
/tae/simp - POST	Obtain annotations and simplifications	input (json)	Json object that specifies the lang, position, text, word

3.3.1.6 Workflow Adaptation Engine

Operation & method	Operation Description	Input	Input Description
/wae/model - GET	Get models matching the specific URI	uri (uri)	The uri of the model
/wae/model - POST	Create a new model	model (json)	Json object containing the new model
/wae/model/page - GET	Get a model with the specific URI and profile	uri (uri)	The uri of the model
		idprofile (string)	The id of the model profile
/wae/model/{objectId} - DELETE	Delete an existing model	objectId (string)	The id of the model that will be deleted
/wae/model/{objectId} - PUT	Update an existing model	objectId (string)	The id of the model that will be updated
		model (json)	The model to be updated

3.3.2 eID

The SIMPATICO platform components and APIs provide functionality that is tailored to the specific end-user. This is the case, for example, for Interactive Frontend, Citizenpedia, Citizen Data Vault, etc. being integrated with the e-services provided by the public administration, the SIMPATICO components should be able to adapt the same end-user authentication and identification mechanisms adopted by the public administration.

The European Commission defines a set of regulations regarding the electronic identity and its use in the internet service market as well as the corresponding interoperability requirements [5]. It requires, in particular, the following criteria for the interoperability framework for the electronic identification and its use within the service market:

- interoperability of national identity schemes
- technology neutral
- common operational security standards

In terms of technical integration and interoperability between the e-services and the SIMPATICO platform, these criteria amount to the following requirements:

- the authentication and user identification mechanism adopted by the SIMPATICO platform should provide a sufficient level of flexibility in order to be used in cooperation with a specific local or national technical solution (provided that the latter adheres to the above mentioned regulations).
- the authentication and access control mechanism adopted by the platform should rely on common security standards in a technology neutral manner.

In order to satisfy these requirements, the following approach has been adopted by the SIMPATICO platform.

First, the authentication and user identification within the platform is performed through a unique point of control, namely the Authentication and Authorization Component (AAC). This component provides the necessary functionality for the user authentication that can be used both for the Web site authentication (necessary, e.g., for Citizenpedia or Interactive Frontend) and for protecting open APIs. The component exposes its functionality using the open security standards, such as OAuth2.0 protocol.

Second, the mechanism decouples the specific local/national implementation of the electronic identity from its use within the SIMPATICO platform. For this purpose, the AAC component provides a protocol- and implementation-neutral model of the identity scheme through the extension points and generic authority management module. More specifically, AAC allows for configuring a specific local electronic identity scheme via a minimal set of user attributes necessary for unambiguous user identification without changing the component implementation. This approach allows for easily integrating a custom authentication provider. In fact, it has already been applied for integrating local solution adopted by the Province of Trento (based on the Smart Card authentication and mobile two-factor authentication), and has been recently extended to the Italian national identity solution (SPID, the eIDAS-compliant electronic identity solution). Furthermore, the AAC component has been

extended to support the common global authentication providers widely used by the 3rd party applications and services, such as Google+ ad Facebook.

The architecture and the flow of activities related to the authentication process is depicted in the following Figure 9.

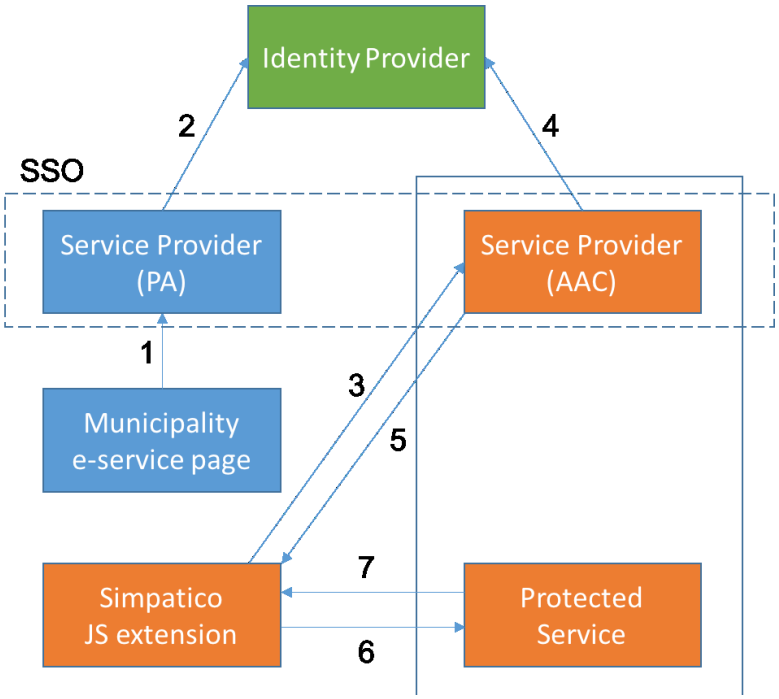

Figure 9: AAC architecture

In this approach, the e-services exposed by the public administration (PA) and the SIMPATICO authentication point (AAC) are federated with the local/national electronic identity provision mechanism. In this way, the two parties operate in a Single Sign-On manner so that the user authenticates only once, and the identity attributes are shared (steps 1-2, 4). It is important to stress, that AAC and consequently SIMPATICO platform components have access only to a minimal set of user attributes and no critical data (e.g., password, signatures, etc.) is compromised. The specific set of attributes is configured through AAC.

Once authenticated, the access to the SIMPATICO components is performed on behalf of the user, following the security flow defined by the OAuth2.0 protocol. That is, the security token is emitted by AAC to the calling party upon explicit user authorization (steps 3-5) and then is used to perform the protected API calls (6-7).

This architecture allows for the necessary steps towards the interoperability of the SIMPATICO platform with respect to the EU regulations. The extensibility of AAC and decoupling from the internal protocol allows for appropriate level of flexibility in order to adhere to specific national technology solutions. Besides, this centralized authentication component adheres to the open and widely-used security standards, which guarantees the necessary level of security assurance.

4 Integration and Deployment Plan within each use-case

In this chapter we represent the activities and steps performed by the use-cases leaders in order to engage the SIMPATICO platform within the citizen e-services. To drive this process, a set of integration scenarios has been defined that, on one side, demonstrate the use cases the platform is required to enable and, on another side, demonstrate how the components of the platform are integrated operationally. These integration scenarios, presented in Section 4.1, are also used to validate that the deployment plan is being accomplished without problems.

The deployment plan, presented in Section 4.2, describes the set of activities to be performed in order to instantiate the SIMPATICO platform. These step include the infrastructure setup and preparation, the software installation activities, and the execution of the integration scenarios.

Finally, the section 4.2.7 to 4.5 describe the specific actions performed by the three use-cases (Trento, Galicia, and Sheffield) on top of the general deployment plan.

4.1 Integration use cases

The Table 7 shows the integration use cases provided in order to test the integration of the whole platform. During the validation of the SIMPATICO platform, one of the activities to be carried out will be the validation of these use cases. All the information will be reported in the Deliverable D5.5[1] at M20 (October 2017).

Table 7: Integration use cases

COMPONENT	USE CASE	SYNOPSIS	COMPONENTS	DEPENDENCIES	PREREQUISITES	BEHAVIOUR	EFFECT
IFE	IFE1	Login	IFE, AAC			Open a service page, press login button. A new window where AAC authentication takes place. Upon window closure, the toolbar is updated and user name is shown	User is authenticated, toolbar unlocked, user name is shown
	IFE2	Session execution	IFE, LOG			User starts/terminates the e-service session	Start/end session events are logged and appear in LOG
	IFE3	E-service form compilation	IFE, LOG			User starts/terminates compilation of the e-service module	Start/end form events are logged and appear in LOG component
	IFE4	Access to an annotated element	IFE, LOG			User access any annotated element (TAE, QAE, CDV)	Click event is logged and appears in LOG component
CDV	CDV1	Link Account to Service	IFE, CDV, AAC	IFE1	e-service is configured in CDV.	Click CDV button in toolbar, CDV popup is shown. Popup asks the user to create a new CDV account and link it to the service	A new account is registered in CDV. Account is linked to the current e-service
	CDV2	Save user data	IFE, CDV	IFE1, CDV1	e-service is configured in	The user fills in the e-service	The user data are saved in

[1] D5.5 SIMPATICO platform validation report v1

					CDV. the user has created a CDV account	form and clicks on the "Save Data" button shown in CDV popup	CDV
	CDV3	Read/populate user data	IFE, CDV	IFE1, CDV1, CDV2	e-service is configured in CDV. the user has already created a CDV account	During the the e-service form compiling, the user can use his/her own data already stored in the CDV selecting them by drop down menu in the requested field	The user can fill in the e-service form retrieving the data in the CDV, previously saved
TAE	TAE1	Free text simplification	IFE,TAE,LOG	IFE1		Select a phrase and click on "Free text simplification" toolbar button. A popup appears, tabs show enriched text.	Each tab show content from TAE. Phrase simplification event is registered in LOG component
	TAE2	Free word simplification	IFE,TAE,LOG	IFE1		Select a single word and click on "Free text simplification" toolbar button. A popup appears, tabs show enriched word.	Corresponding tab show content from TAE. Word simplification event is registered in LOG component
	TAE3	Annotated text simplification	IFE,TAE,LOG	IFE1	E-service elements are annotated with a specific CSS class	The e-service page text elements are annotated. The user clicks the "Text simplification button". The annotated texts are highlighted. Clicking on decoration the simplified versions of text appear.	Text is decorated with enrichment elements. Paragraph simplification event is registered in LOG component
WAE	WAE1	Workflow simplification	IFE,WAE,LOG	IFE1	E-service workflow model is uploaded to WAE repository, form is annotated with the model URI value	Click "workflow adaptation" button. The simplified process starts (the page scrolls to the first workflow block, the block is highlighted, next/prev buttons are added). The user can execute the workflow	The workflow model is downloaded from the WAE repository, the engine starts the model execution. Workflow adaptation request event is registered in LOG component
QAE	QAE1	Login with AAC	QAE,AAC			User signs in using AAC	The user successfully logged in, the profile data is shown in QAE

	QAE2	See e-service questions	IFE,QAE,LOG	IFE1	e-service elements are annotated with a specific class corresponding to e-service paragraph	Click "QAE" toolbar button, the annotated page elements are highlighted. Clicking on the highlighted element shows the questions associated to the paragraph (if any) and a "add a question" link	The list of existing paragraph questions are associated to each annotated paragraph. Upon selecting a specific element, the 'Citizenpedia content request' event is registered in LOG component.
	QAE3	See specific question	IFE,QAE,LOG	IFE1,QAE2		User clicks on a specific question link. A Citizenpedia page with question details is opened in a new window.	Citizenpedia page is opened. The 'Citizenpedia question request' event is registered in the LOG component
	QAE4	Create new question	IFE,QAE,LOG	IFE1,QAE1, QAE2	e-service category is created	Within QAE2 scenario, user clicks the "add a question link". Another browser tab is opened with the precompiled Question form.	QAE is opened with the precompiled form for question creation. Once saved, the question appears in the corresponding paragraph section in IFE. The 'Citizenpedia new question' event is registered in LOG component
	QAE5	Create new answer	QAE,LOG	QAE1		Inside Citizenpedia, the user provides an answer to an existing question.	The number of answers associated to the question is incremented, the answer is saved in the DB. The 'new answer' event is registered in the LOG component
SF	SF1	SF is invoked	SF, IFE		A session is complete	The front-end side of the SF captures the appropriate event of the IFE and asks the back-end of the SF to display a feedback capture form.	The SF feedback form is presented in the front-end for the user to fill in their opinions.
	SF2	SF stores user feedback	SF, LOG	SF1	The user presses the 'Send' button in the SF UI.	The data that the user produces is compiled and stored as the current session's feedback in the LOG.	The LOG is updated with data from the last session.
	SF3	SF generates an appropriate	SF, LOG	SF1	The SF is invoked	The SF internal logic retrieves relevant data from the last	The SF front-end part receives the requested form.

		feedback form				session (e.g., was the text adaptation invoked at any stage) and designs an appropriate feedback form which is sent to the front-end part.	
CPD	CPD1	Login with AAC	CPD, AAC			User signs in using AAC	The user successfully logs in, the profile data is shown in CPD. Depending on the user role (citizen/civil servant) access to some functionality can be denied
	CPD2	Open an existing procedure diagram	CPD		The user has successfully logged in	The user opens a diagram from a list of available (previously created) administrative procedures	The user is presented with a graphical representation of the selected procedure
	CPD3	View existing questions	CPD, QAE		The user has opened an existing procedure diagram	The user selects to view the questions posted on a procedure's activity	The user gets re-directed to the QAE, where they are presented with all the questions related to that activity
	CPD4	Submit a new question	CPD, QAE		The user has opened an existing procedure diagram	The user selects to submit a new question on a procedure's activity	The user gets re-directed to the QAE, where they are provided with a text editor to write a new question related to that activity
	CPD5	Submit a feedback	CPD		The user has opened an existing procedure diagram	The user selects to submit a new feedback (suggestion, comment) on a procedure's activity	The user is presented with a pop-up editor to post a new feedback related to that activity
	CPD6	Create a new procedure diagram	CPD		The user has successfully logged in and is a civil servant	The user selects to create a new procedure diagram	The user is presented with a canvas and a set of tools to draw the procedure phases and describe them with textual content
EE/ DB	EE1	DB presents data	DB, EE	EE4	There is a previously generated report by the EE scheduler.		

	EE2	CPD integration	EE, CPD		The CPD API is available.	The EE back-end accesses the Collaborative Procedure Designer API to gather relevant data (e.g., e-service procedure SVG diagram).	The EE back-end obtains the required data.
	EE3	Citizenpedia integration	EE, CTZP		The CTZP API is available.	The EE back-end accesses the Citizenpedia API to gather relevant data (e.g., statistics of usage per paragraph).	The EE back-end obtains the required data.
	EE4	EE generates a report	EE	EE2, EE3	There is available data from past interactions in the LOG.	Upon invoking from the scheduling mechanism (e.g., cron) the EE back-end retrieves data from the available APIs (LOG, CPD and CTZP) and constructs a report to be displayed in the EE tab of the DB.	A EE report is stored internally for display in future invoking of the DB.
DA	DA1	Data Analysis is invoked on schedule	DA		There exists some user interaction data in the LOG that can be processed.	The scheduling mechanism (e.g., cron) invokes the DA upon its required schedule (e.g., each minute, each hour). The DA executes its internal rules and data available in the LOG which hasn't been processed before is processed.	Aggregated processed data is stored in the LOG for further analysis by DA or representation in the EE

4.2 Deployment of SIMPATICO solution

After the development of each SIMPATICO components, their integration and the set-up of a single SIMPATICO platform, a common deployment plan was designed to configure the technical environment in each Use-case. The Table 8 shows all the activities performed to set-up and launch the three Use-cases and make them ready for the first phase of experimentation.

Table 8: SIMPATICO Common Deployment Plan

Seq. ID	Activity Name	Activity Description	Start Date	Expected Duration	Dependent Activities	Responsible	Involved Team
1	Physical Infrastructure setup	Preparation of the physical infrastructure and the related VM where installing the SIMPATICO components. Check the resource requested by each component	03/04/2017	1	-	Pilot Leader	City
2	Security settings	Configuration of the network taking into account the requirements needed to each platform component	03/04/2017	1	1	Pilot Leader	City
3	AAC installation and configuration	Installation and configuration of the Authentication and Authorization Controller	03/04/2017	1	1-2	Pilot Leader	City
4	CDV installation and configuration	Installation and configuration of the Citizen Data Vault, taking into account the requirements provided by each municipality in order to have the CDV within their network perimeter	03/04/2017	10	1-2-3	Pilot Leader	City ENG
5	IFE installation and configuration	Installation and configuration of the Interactive Front-End. In order to test the component JSclients the corresponding components should be online	03/04/2017	10	1-2-3-4	Pilot Leader	DEUSTO
6	IFE-CDV integration check	Performing the tests to check the integration between the two components	10/04/2017	5	1-2-3-4-5	Pilot Leader	DEUSTO ENG
7	TAE installation and configuration	Installation and configuration of the Text Adaptation Engine	03/04/2017	1	1-2-3	Pilot Leader	FBK USFD
8	WAE installation and configuration	Installation and configuration of the Workflow Adaptation Engine	03/04/2017	1	1-2-3	Pilot Leader	FBK

9	IFE-TAE integration check	Performing the tests to check the integration between the two components	03/04/2017	1	1-2-3-7	Pilot Leader	DEUSTO FBK USFD
10	IFE-WAE integration check	Performing the tests to check the integration between the two components	03/04/2017	1	1-2-3-8	Pilot Leader	DEUSTO FBK
11	LOG installation and configuration	Installation and configuration of the LOG and its related DB	03/04/2017	5	1-2-3	Pilot Leader	HIB
12	SF installation and configuration	Installation and configuration of the Session Feedback	03/04/2017	5	1-2-3	Pilot Leader	HIB
13	DA installation and configuration	Installation and configuration of the Data Analysis	28/04/2017	1	1-2-3	Pilot Leader	HIB
14	EE installation and configuration	Installation and configuration of the Enrichment Engine and Dashboard	28/04/2017	1	1-2-3	Pilot Leader	HIB
15	IFE-SF integration check	Performing the tests to check the integration between the two components	28/04/2017	2	1-2-3-5-12	Pilot Leader	DEUSTO HIB
16	IFE-LOG integration check	Performing the tests to check the integration between the two components	28/04/2017	3	1-2-3-5-11	Pilot Leader	DEUSTO HIB
17	DA-LOG integration check	Performing the tests to check the integration between the two components	28/04/2017	2	1-2-3-11-13	Pilot Leader	DEUSTO HIB
18	EE-LOG integration check	Performing the tests to check the integration between the two components	28/04/2017	2	1-2-3-11-14	Pilot Leader	DEUSTO HIB
19	TAE-LOG integration check	Performing the tests to check the integration between the two components	28/04/2017	3	1-2--3-7-11	Pilot Leader	FBK USFD HIB
20	WAE-LOG integration check	Performing the tests to check the integration between the two components	28/04/2017	3	1-2-3-8-11	Pilot Leader	FBK HIB
21	CTZ installation and configuration	Installation and configuration of the Citizenpedia. In particular, - DB for Collective Knowledge - Q&A Engine	03/04/2017	1	1-2-3	Pilot Leader	DEUSTO BENG

		- Collaborative Procedure Design					
22	IFE-QAE integration check	Performing the tests to check the integration between the two components	10/04/2017	2	1-2-3-5-21	Pilot Leader	DEUSTO
23	IFE-CPD integration check	Performing the tests to check the integration between the two components	28/04/2017	2	1-2-3-5-21	Pilot Leader	DEUSTO BENG
24	EE-CPD integration check	Performing the tests to check the integration between the two components	28/04/2017	2	1-2-3-14-21	Pilot Leader	HIB BENG

4.2.1 Infrastructure Requirements and Setup

Deployment of the SIMPATICO platform amounts to installation and integration of a set of the software components that provide the required APIs or Web user interface towards citizens or civil servants.

The recommended environment for the platform installation is described in the following table:

Operating System	Linux	No specific constraints.
Execution Environment	JVM Python Node.js Apache Tomcat Elasticsearch	1.8+ 2.7+ 4.2.3+ 8+ 5.x
Database	MySQL MongoDB	5.5+ 3.4+
Storage	> 100GB	Depends on TAE configuration
RAM	> 30GB	Depends on TAE configuration
Processors	> 8CPU	Depends on TAE Configuration

The required resources may vary depending on the configuration used by the Text Adaptation Engine. More specifically, the amount of RAM, storage, and processing depends on the language(s) engaged by the specific platform instance.

For the authentication and access control, the SIMPATICO platform components rely on the SIMPATICO AAC component, which provides the support for OAuth2.0 authentication protocol and serves as a authorization server for accessing the platform resources and APIs. It also intermediates the identity provisioning tasks, allowing for integration with the city-specific or national-level authentication systems through its modular architecture. The AAC component is deployed as a Java Web application in a standard JEE container (e.g., Apache Tomcat). It provides the out-of-the-box integration of the Google and Facebook authentication systems.

Detailed description of the installation and setup instructions are presented here:

https://github.com/SIMPATICOProject/aac

It is strongly recommended that the access to the platform APIs and components is protected using HTTPS connection as some of the user-related information is being passed between the user fronted (IFE) and the server components (e.g., CDV).

4.2.2 CDV Installation and Setup

Citizen Data Vault exposes the functionality of the citizen personal data repository. This functionality is exposed via API, specifically RESTful services. CDV setup requires the following environment:

- MySQL 5.7+, MongoDB 3.3+
- Java 1.8+
- Apache Tomcat 8+

The detailed installation and setup procedure is described here:

https://github.com/SIMPATICOProject/CDV

Please note that for the city-specific e-services to be integrated with CDV, it is necessary to populate the CDV database with the mapping of the e-service fields and the user profile data fields. This mapping should be provided via CDV API at the moment of the e-service configuration.

4.2.3 TAE and WAE Installation and Setup

Text Adaptation Engine and Workflow Adaptation Engine expose advanced techniques for improving the use of the e-services by citizens and the interactions with those services. The engine setup consists of server-side components installation and a front-end configuration as a part of the Interactive Front-End setup. As for the server-side components, the specific procedure depends on the languages used and may vary. The setup requires the following environment:

- MongoDB 3.4+
- Java 1.8+, Python 2.7+ (for English and Spanish setup)

More detailed installation instructions may be found here:

https://github.com/SIMPATICOProject/simpatico-adaptation-engines

Please note that the workflow models of the e-services deployed by the city should be stored in the WAE repository using the corresponding APIs. The details about the workflow model and its usage can be found here:

https://github.com/SIMPATICOProject/simpatico-adaptation-engines/blob/master/doc/wae-model.docx

4.2.4 LOG, SF, and DA Installation and Setup

Interaction Log, Data Analysis, and Session Feedback components provide a way to collect, aggregate, and analyse the user interaction data through a series of APIs. As in case of other components, the setup amounts to the installation of server-side component and IFE configuration. The required environment for these tools include:

- Elasticsearch 5x
- Java 1.8+
- Apache Tomcat 8+

The installation insturctions for the interaction log components may be found here:

https://github.com/SIMPATICOProject/logs

4.2.5 Enrichment Engine and Dashboard Installation and Setup

Enrichment engine and dashboard allow for the analysis and visualization of the interaction logs collected through Interactive Front End. The enrichment engine is realized as a set of scripts that are periodically executed on top of the interaction log APIs in order to elaborate more complex and

aggregated data. The Dashboard provides the civil servants with the possibility to visualize those data. Besides, the Dashboard provides a tool for the visualization and analysis of the complexity of texts used by e-services.

The execution environment required by the components include:

- Python 2.7+ (for EE), Java 1.8+ (for dashboard)
- Apache Tomcat 8+ (for dashboard)
- Interactive Logs and TAE installed

The detailed instructions for the EE installation and setup can be found here:

https://github.com/SIMPATICOProject/ee

The detailed instructions for the Dashboard installation and setup can be found here:

https://github.com/SIMPATICOProject/dashboard

4.2.6 Citizenpedia Installation and Setup

Question and Answer Engine (QAE) provides the user with a knowledge base, where the citizen can ask and discuss the issues related to the specific e-services. The Collaborative Process Designer (CPD) is a tool for visualizing the administrative procedures overarching the e-services provided by a city. Together these tools form SIMPATICO Citizenpedia.

The execution environment required bt the components include:

- Node.js 4.2.3+ (for QAE) and Java 1.8+ (for CPD)
- MongoDB 3.4+

The detailed instructions for the QAE installation and setup can be found here:

https://github.com/SIMPATICOProject/citizenpedia

The detailed instructions for the QAE installation and setup can be found here:

https://github.com/SIMPATICOProject/CPD

Please note that the e-service descriptions may be done from scratch or starting from the test data embedded in the CPD installation.

4.2.7 IFE and e-Service configuration

Interactive front-end (IFE) represents an extension to be added to e-service Web page in order to enable the access to the SIMPATICO functionality. This extension is implemented as a set of JavaScript files and other assets (e.g., images, CSS, etc) that provide the user with the UI tools to access text adaptation, workflow adaptation, CDV, Citizenpedia, and session feedback.
The source code of the IFE modules with the corresponding setup instructions and examples can be found here:

https://github.com/SIMPATICOProject/IFE

The standard IFE configuration assumes that the SIMPATICO tools are made available via a fixed toolbar, where a set of buttons provides access to the specific tool functionality. The toolbar is made available to authenticated users only, hence the integration with AAC for user identification and

authentication. Please note that the combination of the tools as well as the visualization of the toolbar may be customized via the simpatico-ife.js file, where it is possible to change the way the individual components are activated and structured. The appearance of the toolbar and of the individual tools may be customized via CSS changes.

The standard integration of the SIMPATICO platform with an e-service consists of the following steps:

I. Set up and configure the SIMPATICO environment

The IFE is integrated into the e-service page via importing a set of JavaScript and CSS files required by the UI tools. It is also necessary to configure a set of global properties specific to the current e-service page.

```
<!-- SIMPATICO BEGIN -->
<link rel="stylesheet" type="text/css" href="css/simpatico.css" />

<link rel="stylesheet"
       href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css" />
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

<script type="text/javascript">
      var simpaticoEservice = "2"; //  the id corresponding to the e-service
      var simpaticoCategory = "Infanzia"; // the general category of the e-service
                                                                                var
simpaticoMapping=["AventeTitolo_EMailPEC","AventeTitolo_Fax","AventeTitolo_EMail"];
      var ERROR_LABELS = {
        'block1' : 'Manca il codice fiscale',
        'block4' : 'Manca selezione Part-time / Full-time'
      }
</script>

<script src="./js/ctz-ui.js"></script>
<script src="./js/ctz-core.js"></script>
<script src="./js/cdv-ui-popup.js"></script>
<script src="./js/cdv-core-popup.js"></script>
<script src="./js/tae-core.js"></script>
<script src="./js/tae-ui.js"></script>
<script src="./js/wae-core.js"></script>
<script src="./js/wae-ui.js"></script>
<script src="./js/tae-core-popup.js"></script>
<script src="./js/tae-ui-popup.js"></script>
<script src="./js/simpatico-auth.js"></script>

<script src="./js/simpatico-ife.js"></script>

<!-- SIMPATICO END -->
```

Besides, the individual tools may require specific configuration that are achieved via HTML5 custom attributes or through a set of custom CSS classes as shown below.

II. Set up Workflow Adaptation SIMPATICO component binding

First, it is necessary to bind the form to the corresponding workflow model:

```
<form data-simpatico-workflow="http://simpatico.eu/test" …>
```

Second, it is necessary to markup the definition of the workflow blocks. More specifically, each workflow block is identified by the HTML class element “simpatico-block”. The workflow block definition can be done in two ways.

a) if the workflow block matches with an HTML element, the class parameter of this element must be

overloaded with the value “simpatico-block” (e.g. <table class=”simpatico-block …”>).

```
<table class="simpatico-block …" …>

…
</table>
```

b) If the workflow block matches with more than one HTML elements, a new element surrounding the elements set must be added (e.g.).

```
<div class="simpatico-block">
        <p …>…</p>
        <table …>…</table>
        <table …>…</table>

</div>
```

III. Set up the Question and Answer SIMPATICO component binding

```
<table class="simpatico-question-and-answer …" …>

    …
</table>

or

<div class="simpatico-question-and-answer">
        <p …>…</p>
        <table …>…</table>
        <table …>…</table>
</div>
```

IV. Set up the Text Adaptation SIMPATICO component binding

```
<p class="simpatico-text-paragraph">…</p>
```

In the following three sections each Use-case describes what is the configuration and deployment activities specific for their City.

4.3 Trento use-case

Trento pilot adopts the standard SIMPATICO integration approach for all components but the Citizen Data Vault. The e-services HTML pages have been enriched with the required Javascript and tags in order to enable the SIMPATICO toolbar. The requirement to integrate the CDV with a different approach comes directly as a constraint from the Municipality o Trento.

The Municipality of Trento has adopted “Sportello Telematico”, an end-to-end solution provided by company "GLOBO srl", specifically targeting the digitalization of modules for service provision by public administrations. Within this solution, the digital module is a composition of sections of organic information (e.g., birth data section, residence data section, real estate registry data section). The

module designer explicitly maps the logic of the interaction with an information section. The integrations with legacy systems are handled via a centralized REST web service, which routes the proper service request to the right data source service. Finally the solution supports module hierarchy, which guarantees the definition of a well-organized digital module library.

“Sportello Telematico” supports operations on the fields, e.g., pre-filling their values, via explicit external calls. These calls are handled via a unique REST web service, which is responsible of routing the proper request to the proper external services and data source and to compose the reply. The web service request and response must implement a specific simple grammar: the request message is formatted in XML and contains an array of key-value couple (request params); the response message is formatted in XML and contains an array of key-value couple (relations) or a set of key-value couples (record).

The integration uses two different strategy:

1) Injection of JavaScript in the different digital modules;
2) Invocation of REST web services to handle interaction with legacy systems.

The injection of JavaScript in the digital module is possible only if it does not interfere with the module interaction logic. This means that the injected JavaScript can operate on the DOM modifying only the static elements of the document (in particular, the labels and text descriptions), but cannot operate in the fields that the user shall fill. This strategy has been used in order to integrate SIMPATICO Interactive FrontEnd (IFE) component.

For what concerns the integration with the local IT systems, we deployed the CDV component within the borders of the information system of the Comune di Trento, while the others SIMPATICO tools can be deployed on the project cloud infrastructure. This is mainly due to the constraints posed the fact the CDV contains user personal data. This solution will prevent voluntary or accidental access to the personal data. This solution simplifies the integration security and authentication issues between “Sportello Telematico” and CDV.

4.4 Galicia use-case

The Galicia pilot models a potential deployment of two e-services for the wellness of the citizens:

- BS607A regarding elderly Wellness and SPAs (therapeutic and medicinal baths) (original e-service: https://sede.xunta.gal/detalle-procedemento?codtram=BS607A)
- BS613B regarding the fostering of autonomy for the citizens with disabilities (original e-service: https://sede.xunta.gal/detalle-procedemento?codtram=BS613B)

On the goals level, the deployment follows objectives along the same lines as other pilots: simplifying complex texts and procedures for users falling in the particular details of the demographic context in Galicia, such as a predominantly elder population and commonly a rural environment.

Technically the pilot presented some particular difficulties. The real e-services are offered integrated in the overall Xunta de Galicia platform for electronic administration offered by their technological branch AMTEGA[2]. Integration with the real platform was examined but quite involved and presenting significant risks for the public administration (the platform has region-wide deployed services) so after consideration with the responsible parties and the project management a middle way for piloting the SIMPATICO demonstrators was chosen.

[2] http://amtega.xunta.gal/ - AMTEGA website (Spanish and Galician only)

A visually complete replica of both e-services was constructed. The original e-services were comprised of a HTML landing and information page followed by the completion of editable PDF forms which were then securely submitted. The platform had in place user authentication leveraging on the smartcard-based national ID card.

These facilities have been replaced by SIMPATICO components. Authentication is now based on the AAC component and the front-end is substituted by a full HTML replica of the PDF forms. This replication makes for straightforward integration of the rest of the components. The IFE can use Javascript injection upon the replica sites without major problems.

The replica front-ends and the rest of the components are deployed locally in a server located at the HI Iberia premises. This is a high performance HP ProLiant server with full virtualization of the environments. A production and a development environments are deployed simultaneously and can run in parallel with no interference. External access to the servers is configured so the developers from other SIMPATICO partners can access for installation and testing as well as eventually the citizens from Galicia participating in the evaluation tests.

4.5 Sheffield use-case

The main objective of Sheffield Pilot is to improve the interaction between citizens and PA and to deliver more simplified services to their Citizens. Sheffield City Council (SCC) has adopted Adobe solution and published their new website with better User interface to achieve the same.

The goal behind incorporating SIMPATICO functionality in the legacy system is to deliver simplified and user friendly services to their citizens. Sparta as Technical partner is using following strategy to integrate SIMPATICO in the sandbox environment of legacy system.

The integration uses two different strategies:

1) Injection of JavaScript in the Adobe Experience Manager (AEM) template
2) Invocation of REST web services to handle interaction between SIMPATICO components and the AEM

Injection of Javascript code (i.e., IFE component) is carried out directly in the AEM files and rest of the components have been installed using Docker system. The interaction between components located in separate Dockers and the Javascript framework has been configured via Rest APIs. It carried out in same way with the AEM on which the prototype of SCC website is running.

Currently the Sandbox environment is running at Sparta's server, where the AEM and the latest prototype of SCC's website were also deployed. Docker[6] approach has been adopted in deploying the different components in separate containers for rapid deployment and simplified configuration. REST web services have been used to configure the interaction between components and the interaction between SIMPATICO and test legacy system.

For the security concerns of User data, the plan is to deploy the solution of Simpatico locally to Sparta's server, validated and tested by dummy users and once things will arrange in order, the working solution will be moved to SCC's server mainly due to the constraints bound to the fact the CDV contains user personal data.

5 Conclusion

In this document we have presented what have been the activities carried out to integrate and deploy the SIMPATICO components. In particular, we have taken into account some **guidelines** given by the European Interoperability Framework regarding the semantic and technical interoperability; we have described the **OpenAPIs** provided by each component and the integration of the SIMPATICO **authentication system** with the municipality one. In the end, we have developed a **Deployment Plan** and described the **integration use cases** to be validated during the test of the whole platform.

Moreover, at the end of March 2017, ISA^2 has delivered a new release of European Interoperability Framework (Figure 10) that has focused more **emphasis on how interoperability principles and models should apply in practice**. The number of recommendations has increased from 25 to 47[7]. Since we have not enough time to study and understand this new version of EIF, in the second version of this deliverable the new release of EIF will be investigated and the SIMPATICO Interoperability Framework will take into account the improvement and the new guidelines proposed by ISA^2 team.

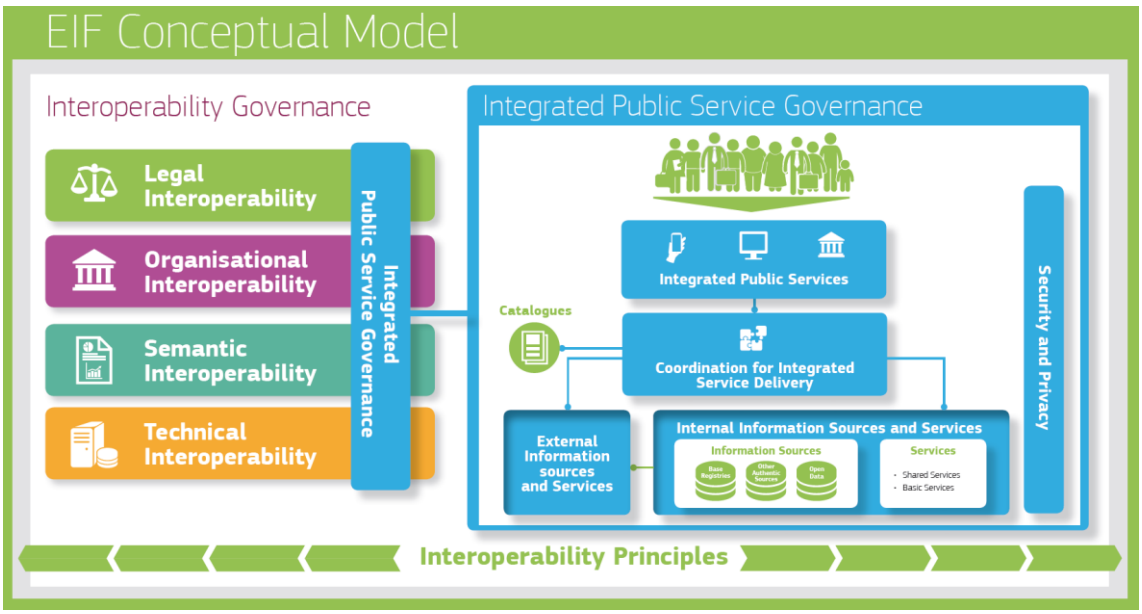

Figure 10: New European Interoperability Framework (source: ISA^2 [7])

In the same way, after the first running phase of the SIMPATICO use-cases all the **feedback coming from the SIMPATICO e-services users** will be collected and analysed to generate improvement to the platform, its set-up and deployment.

6 References

[1] E. Communities, “European Interoperability Framework For Pan-European Egovernment Services,” 2004.

[2] C. d. online, 2017. [Online]. Available: http://www.computer-dictionary-online.org/index.asp?q=interoperability. [Accessed 21 March 2017].

[3] R. C. H. Kubicek, “Interoperability in eGovernment: A Survey on Information Needs of Different EU Stakeholders,” *European Review of Political Technologies,* 2005.

[4] Swagger, “swagger.io,” 2016. [Online]. Available: http://swagger.io/. [Accessed 27 March 2017].

[5] European Parliament and Council, “Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC,” 2014.

[6] “What is Docker,” [Online]. Available: https://www.docker.com/what-docker. [Accessed 30 04 2017].

[7] ISA2, “New European Interoperability Framework,” [Online]. Available: https://ec.europa.eu/isa2/eif_en. [Accessed 18 04 2017].

[8] C. C. SPC, “Linee guida per l’interoperabilità semantica attraverso i linked open data,” 2012.