SALUS
“Scalable, Standard based Interoperability Framework for Sustainable Proactive Post Market Safety Studies”

SPECIFIC TARGETED RESEARCH PROJECT

PRIORITY Objective ICT-2011.5.3 b) Tools and environments enabling the re-use of electronic health records

SALUS D6.3.1 Integrated SALUS Components – R1

Due Date: July 31, 2013
Actual Submission Date: July 29, 2013
Project Dates: Project Start Date: February 01, 2012
Project End Date: January 31, 2015
Project Duration: 36 months
Deliverable Leader: SRDC

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>

X
### Document History:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
<th>From</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1</td>
<td>2013-07-05</td>
<td>Initial Document</td>
<td>SRDC</td>
<td>All Consortium</td>
</tr>
<tr>
<td>v0.2</td>
<td>2013-07-10</td>
<td>Filled the INSERM related sections in</td>
<td>INSERM</td>
<td>SRDC</td>
</tr>
<tr>
<td>v0.3</td>
<td>2013-07-10</td>
<td>Filled the OFFIS related sections in</td>
<td>OFFIS</td>
<td>SRDC</td>
</tr>
<tr>
<td>v0.4</td>
<td>2013-07-16</td>
<td>SIL and Semantic Services</td>
<td>SRDC</td>
<td>Internal</td>
</tr>
<tr>
<td>v0.5</td>
<td>2013-07-17</td>
<td>Terminology Server, Security &amp; Privacy</td>
<td>SRDC</td>
<td>Internal</td>
</tr>
<tr>
<td>v0.6</td>
<td>2013-07-22</td>
<td>Overall integration, content formatter, SIL</td>
<td>SRDC</td>
<td>Internal</td>
</tr>
<tr>
<td>V0.7</td>
<td>2013-07-24</td>
<td>Executive Summary, Introduction, Conclusion</td>
<td>SRDC</td>
<td>All Consortium</td>
</tr>
</tbody>
</table>
### Contributors (Benef.)

Gokce B. Laleci (SRDC), Ali Anil Sinaci (SRDC), Suat Gonul (SRDC), Gunnar Declerck (INSERM), Frerk Muller (OFFIS)

<table>
<thead>
<tr>
<th>Responsible Author</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suat Gonul</td>
<td><a href="mailto:suat@srdc.com.tr">suat@srdc.com.tr</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beneficiary</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRDC</td>
<td>+90-312-2101763</td>
</tr>
</tbody>
</table>
## SALUS Consortium Contacts:

<table>
<thead>
<tr>
<th>Beneficiary</th>
<th>Name</th>
<th>Phone</th>
<th>Fax</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRDC</td>
<td>Gokce Banu Erturken</td>
<td>+90-312-2101763</td>
<td>+90(312)2101837</td>
<td><a href="mailto:gokce@srdc.com.tr">gokce@srdc.com.tr</a></td>
</tr>
<tr>
<td>EUROREC</td>
<td>Georges De Moor</td>
<td>+32-9-2101161</td>
<td>+32-9-3313350</td>
<td><a href="mailto:georges.demoor@ugent.be">georges.demoor@ugent.be</a></td>
</tr>
<tr>
<td>UMC</td>
<td>Niklas Norén</td>
<td>+4618656060</td>
<td>46 18 65 60 80</td>
<td><a href="mailto:niklas.noren@who-umc.org">niklas.noren@who-umc.org</a></td>
</tr>
<tr>
<td>OFFIS</td>
<td>Wilfried Thoben</td>
<td>+49-441-9722131</td>
<td>+49-441-9722111</td>
<td><a href="mailto:thoben@offis.de">thoben@offis.de</a></td>
</tr>
<tr>
<td>AGFA</td>
<td>Dirk Colaert</td>
<td>+32-3-4448408</td>
<td>+32 3 444 8401</td>
<td><a href="mailto:dirk.colaert@agfa.com">dirk.colaert@agfa.com</a></td>
</tr>
<tr>
<td>ERS</td>
<td>Gerard Freriks</td>
<td>+31 620347088</td>
<td>+31 847371789</td>
<td><a href="mailto:g.freriks@e-Recordservices.eu">g.freriks@e-Recordservices.eu</a></td>
</tr>
<tr>
<td>LISPA</td>
<td>Alberto Daprá</td>
<td>+390239331605</td>
<td>+39 02 39331207</td>
<td><a href="mailto:alberto.dapra@lispa.it">alberto.dapra@lispa.it</a></td>
</tr>
<tr>
<td>INSERM</td>
<td>Marie-Christine Jaulent</td>
<td>+33142346983</td>
<td>+33153109201</td>
<td><a href="mailto:marie-christine.jaulent@crc.jussieu.fr">marie-christine.jaulent@crc.jussieu.fr</a></td>
</tr>
<tr>
<td>TUD</td>
<td>Peter Schwarz</td>
<td>+49 351 458 2715</td>
<td>+49 351 458 7319</td>
<td><a href="mailto:Peter.Schwarz@uniklinikum-dresden.de">Peter.Schwarz@uniklinikum-dresden.de</a></td>
</tr>
<tr>
<td>ROCHE</td>
<td>Jamie Robinson</td>
<td>+41-61-687 9433</td>
<td>+41 61 68 88412</td>
<td><a href="mailto:jamie.robinson@roche.com">jamie.robinson@roche.com</a></td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

This document describes the integration activities carried out as a result of the first development cycle of SALUS project. The integration of the components developed in the first development cycle of SALUS project are described in this deliverable. These include:

- Integration of Terminology Server with
  - Case Series Characterization Tool
  - ADE Notification Tool
  - ICSR Reporting Tool

- Integration of Authentication Service with
  - CDE Repository
  - Case Series Characterization Tool (CSCT)

- Integration of De-Identification Service with
  - ICSR Reporting Tool

- Integration of Safety Analysis Query Manager with
  - Case Series Characterization Tool

- Integration of Semantic Interoperability Layer Data Service with
  - Safety Analysis Query Manager
  - ADE Notification Tool
  - ICSR Reporting Tool
  - Content Formatter

- Integration of Technical Interoperability Layer with
  - Semantic Interoperability Layer Data Service

- Integration of ADE Notification Tool with ICSR Reporting Tool

In addition to these, we also report on the additional interfaces already implemented to be used by the phase II components and the planned integration activities in the second development cycle briefly.
# TABLE OF CONTENTS

**Executive Summary** .......................................................................................................................... 5

**Table of contents** ............................................................................................................................... 6

1 **INTRODUCTION** ............................................................................................................................. 11

1.1 Purpose ........................................................................................................................................... 7

1.2 Scope and Relationship with other SALUS Deliverables ............................................................... 7

1.3 Abbreviations and Acronyms ......................................................................................................... 8

1.4 An Overview of Integration activities in the first development cycle ........................................... 9

2 **INTEGRATION OF SALUS COMPONENTS** ................................................................................. 11

2.1 Integration with Terminology Server ............................................................................................. 11

2.1.1 Case Series Characterization Tool ............................................................................................ 16

2.1.2 ADE Notification Tool ............................................................................................................. 16

2.1.3 ICSR Reporting Tool ............................................................................................................... 17

2.2 Integration with Security & Privacy Services ................................................................................ 18

2.2.1 Integration with Authentication Services ................................................................................... 18

2.2.2 Integration with Deidentification – Pseudonymization Services ............................................ 19

2.2.2.1 ICSR Reporting Tool ........................................................................................................... 20

2.2.2.2 Patient History Tool ............................................................................................................ 21

2.2.2.3 Post Market Safety Study Tool ............................................................................................ 21

2.2.3 Integration with Audit Services ................................................................................................ 21

2.3 Integration with Semantic Services .............................................................................................. 24

2.3.1 Integration with Safety Analysis Query Manager ................................................................. 25

2.3.1.1 Case Series Characterization Tool ....................................................................................... 25

2.3.1.2 Post Market Safety Study Tool ........................................................................................... 26

2.4 Integration with Semantic Interoperability Layer ........................................................................ 26

2.4.1 Integration with Semantic Interoperability Layer Data Service .............................................. 26

2.4.1.1 Safety Analysis Query Manager .......................................................................................... 27

2.4.1.2 ADE Notification Tool ....................................................................................................... 27

2.4.1.3 ICSR Reporting Tool ......................................................................................................... 28

2.4.1.4 OMOP Converter ................................................................................................................ 28

2.4.2 Integration with Content Formatter ......................................................................................... 28

2.4.2.1 Semantic Interoperability Layer Data Service ................................................................... 29

2.5 Integration with Technical Interoperability Layer ....................................................................... 29

2.5.1 Semantic Interoperability Layer Data Service ....................................................................... 34

2.6 ANT – IRT integration .................................................................................................................... 36

3 **CONCLUSION & FUTURE WORK [SRDC]** ................................................................................. 37
1 INTRODUCTION

1.1 Purpose
The purpose of this deliverable is to report the integration activities carried out as a result of the first development cycle of SALUS project.

1.2 Scope and Relationship with other SALUS Deliverables
Based on our Description of Work, SALUS development activities are being carried out in two cycles. As presented in Figure 1, the first development cycle is carried out between Month 9 and Month 18.

Initial versions of almost all of SALUS software components have been delivered within the first 16 months of the project. These components at a high level can be listed as follows:

- Terminology Server (SALUS D4.4.1)
- CDE Repository Tool (SALUS D4.2.1)
- Case Series Characterization Tool (SALUS D6.2.1)
- ADE Notification Tool (SALUS D6.1.1)
- De-Identification and Pseudonymization services (SALUS D5.4.1)
- Semantic Interoperability Layer (SALUS D4.4.1)
  - Semantic Interoperability Layer Data Service
  - Safety Analysis Query Manager
  - Content Formatter Services
  - Converter Services
- Technical Interoperability Layer (SALUS D5.1.1 and D5.2.1)

The integration task in the first development cycle focused on the integration of these tools with each other. In addition to these, while these components are being developed, based on the conceptual design of SALUS architecture (see Deliverable 3.4.1), for each component the external services that
will be opened to other components (to be utilized by not only the phase I components by also by the components to be implemented in the second development cycle) have been designed and implemented. In this deliverable we will focus on presenting the results of the integration activities in the first development cycle. In addition to this we also report on the additional interfaces to be used by the phase II components and the planned integration activities briefly.

The following documents were used or referenced in the development of this document:
- SALUS Description of Work (SALUSPartB_20110118.pdf)
- D3.4.1 Conceptual Design of the SALUS Architecture
- D4.2.1 SALUS Common Set of Data Elements for Post Market Safety Studies – R1
- D4.3.1 SALUS Harmonized Ontology for Post Market Safety Studies – R1
- D4.4.1 SALUS Semantic Mediation Framework – R1
- D5.1.1 Subscription Based Interoperability Profiles and Open Source Toolsets – R1
- D5.2.1 Query Based Interoperability Profiles and Open Source Toolsets – R1
- D5.3.1 Open Source Toolsets for Reporting Activities for Post Market Safety Studies – R1
- D5.4.1 Interoperability Profiles and Open Source Toolsets for Security and Privacy – R1
- D6.1.1 Toolsets for Enabling ADE Detection on EHRs based on temporal patterns – R1
- D6.2.1 Toolsets for enabling Signal detection on EHRs based on temporal patterns – R1

### 1.3 Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation/Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADE</td>
<td>Adverse drug event</td>
</tr>
<tr>
<td>ANT</td>
<td>ADE Notification Tool</td>
</tr>
<tr>
<td>CDE</td>
<td>Common Data Element</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Information Model</td>
</tr>
<tr>
<td>CM</td>
<td>Care Management Profile</td>
</tr>
<tr>
<td>CSCT</td>
<td>Case Series Characterization Tool</td>
</tr>
<tr>
<td>EHR</td>
<td>Electronic health record</td>
</tr>
<tr>
<td>HLGT</td>
<td>High level group term</td>
</tr>
<tr>
<td>HLT</td>
<td>High level term</td>
</tr>
<tr>
<td>ICSR</td>
<td>Individual case safety report</td>
</tr>
<tr>
<td>IRT</td>
<td>ICSR Reporting Tool</td>
</tr>
<tr>
<td>OID</td>
<td>Object identifier</td>
</tr>
<tr>
<td>PHT</td>
<td>Patient History Tool</td>
</tr>
<tr>
<td>PMSST</td>
<td>Post Market Safety Study Tool</td>
</tr>
<tr>
<td>PSN</td>
<td>Pseudonym</td>
</tr>
<tr>
<td>PT</td>
<td>Preferred term</td>
</tr>
<tr>
<td>QED</td>
<td>Query for Existing Data Integration Profile</td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>SAQM</td>
<td>Safety Analysis Query Manager</td>
</tr>
<tr>
<td>SAT</td>
<td>Safety analysis tools</td>
</tr>
<tr>
<td>SIL</td>
<td>Semantic Interoperability Layer</td>
</tr>
<tr>
<td>TAS</td>
<td>Temporal Association Screening</td>
</tr>
<tr>
<td>TIL</td>
<td>Technical Interoperability Layer</td>
</tr>
<tr>
<td>TPC</td>
<td>Temporal Pattern Characterization</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Definition Language</td>
</tr>
</tbody>
</table>
1.4 An Overview of Integration activities in the first development cycle.

Currently at Month 18 of the project lifetime, SALUS is a multi-module project composed of 60 separate modules according to implementation. Most of these modules are implemented using the Java programming language. However, among these 60 modules there are modules dedicated to the safety analysis tools, which are developed in the scope of pilot scenarios of the SALUS project, such as Case Series Characterization Tool, ADE Notification Tool, ICSR Reporting Tool, etc. Including Java Script codes used for the implementation of graphical user interfaces of these tools.

Different SALUS components have been integrated through the following two methodologies:

1. **Direct Java calls**: When two different components are communicating, if none of them requires to be stand-alone processes in the server machines, then the integration is achieved through direct Java API calls. User authentication module under security-privacy is an example of this. While integrating authentication with Case Series Characterization Tool, Java API methods of the AuthenticationManager is used directly by the CSCT. Because, authentication does not need to be a stand-alone process, it only makes an abstraction on the user authentication database and provides convenient methods. Different components using the authentication module should only aware of the changes in the database.

2. **RESTful communication**: If both of the communicating components require to be stand-alone processes in the server machines, then integration is achieved through secure HTTP calls with RESTful service invocations. In the scope of SALUS Project, this type of communication is being used when the components are deployed on two separate physical servers. For instance, in the LISPA deployment, ICSR Reporting Tool and De-Identification Services are deployed on separate physical servers. So, ICSR Reporting Tool benefits from the services, which are served as RESTful services, of De-Identification component through the HTTP protocol. Another usage of this type of integration is between the graphical user interfaces and server side implementations of the safety analysis tools. Server side implementations of safety analysis tools provides RESTful services to be exposed by the graphical user interfaces, that forms the communication between the graphical user interface and server side.

In order to force the integration from the low-level development aspects to the high-level HTTP interactions, SALUS components have been being developed as a big Maven' project. Maven provides compilation of the all of the implementation modules in a single step. In addition to the compilation, this single step also contains execution of several types of tests such as unit tests and integration tests. Especially, the execution of the integration test ensures the integration between several components is satisfied successfully.

In the following box, the root maven module, which provides compilation and testing all of the SALUS components listed in `<module>` tags, can be seen. Some of these components are also root modules having sub-modules. For instance, the `sat` module includes all of the modules composing of all of the safety analysis tools.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

1 http://maven.apache.org/
From the conceptual point of view, another module that is needed to be explained considering the integration aspect is the parent module which can be seen in the above module definition as well. This module contains the third party library dependencies together with their versions. Having such a common module ensures the usage of the same versions of third party libraries e.g. logging libraries by several components, which prevents version conflicts and enables the usage of those components in the same context e.g. in the same web application.

The Figure 2 depicts the integration between the SALUS components together with the correspondent integration type e.g. through Java API or RESTful API. In the first version of this Deliverable we will mainly focus on the components included in the use cases that we have been working till Month 18.
That is, we mainly present the integration of the components whose implementations are finished. Nevertheless, some of the integration points are mentioned conceptually as plans for the next period e.g. integration of OMOP Converter with the SIL-DS. This indicates that the integration of such components will be given in detail in the next versions of this deliverable as we proceed with the implementation of those components.

![High level view of integration of SALUS components](image)

Figure 2 – High level view of integration of SALUS components

In the following sections the integration among these components will be explained in detail.

## 2 INTEGRATION OF SALUS COMPONENTS

### 2.1 Integration with Terminology Server

SALUS has identified a set of terminology systems to be incorporated within its interoperability framework based on the requirements of the EHR sources and end-user organizations. SALUS deliverable “D4.3.1 – Harmonized Ontology for Post Market Safety Studies – R1” presents a list and details of the terminology systems – under the title of SALUS Terminologies – which are semantically addressed within the related SALUS components.

Terminology Server is the associated component of the SALUS architecture which serves the semantic definitions of and the relations (i.e. semantic mappings) between the terminologies. Terminology Server gives its responses following the Linked Data and Semantic Web technologies. That is, the services of the Terminology Server are designed and implemented to give RDF descriptions using the well-accepted thesauri such as SKOS and FOAF.
As of the 1st phase of the SALUS development cycle, the Terminology Server can serve all terminology systems which have been reported in D4.3.1. The terminology systems are identified by their OIDs – Object Identifiers – or their labels so that they can be uniquely identified by the Terminology Server (and by any universal system being aware of the OID specification). A terminology system can have more than one label and these labels can be used in the REST interactions regardless of the letter cases. The terminology systems in use can be listed as follows:

- **ATC**: The Anatomical Therapeutic Chemical Classification System, WHO Collaborating Centre for Drug Statistics Methodology (WHOCC)
  o **OID** = 2.16.840.1.113883.6.73
  o **Labels** = \{ATC\}

- **MedDRA**: Medical Dictionary for Regulatory Activities Terminology, Version 13.0
  o **OID** = 2.16.840.1.113883.6.163
  o **Labels** = \{MDR, MedDRA\}

- **SNOMED-CT**: Systematized Nomenclature of Medicine--Clinical Terms
  o **OID** = 2.16.840.1.113883.6.96
  o **Labels** = \{SNOMED CT, SNOMEDCT\}

- **ICD-10**: International Statistical Classification of Diseases and Related Health Problems. 10th rev.
  o **OID** = 2.16.840.1.113883.6.3
  o **Labels** = \{ICD-10, ICD10\}

- **ICD-10-GM**: Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme, 10. Revision, German Modification
  o **OID** = ICD-10-GM has no OID. Therefore, in order to operate on all terms belonging to different versions of ICD-10-GM, terminology server should be queried with a label. If an OID of the ICD-10-GM versions is given (i.e. OID of icd10gm2004), the only the terms of that version is included in the queries.
  o **Label** = \{icd10gm\}
  o The German modification of ICD-10 has several versions as indicated below together with their OIDs:
    - icd10gm2004
      o **OID** = 1.2.276.0.76.5.302
      o **Labels** = \{icd10gm2004\}
    - icd10gm2005
      o **OID** = 1.2.276.0.76.5.304
      o **Labels** = \{icd10gm2005\}
    - icd10gm2006
      o **OID** = 1.2.276.0.76.5.311
      o **Labels** = \{icd10gm2006\}
    - icd10gm2007
      o **OID** = 1.2.276.0.76.5.318
      o **Labels** = \{icd10gm2007\}
    - icd10gm2008
      o **OID** = 1.2.276.0.76.5.330
      o **Labels** = \{icd10gm2008\}
    - icd10gm2009
      o **OID** = 1.2.276.0.76.5.356
      o **Labels** = \{icd10gm2009\}
    - icd10gm2010
      o **OID** = 1.2.276.0.76.5.384
      o **Labels** = \{icd10gm2010\}
    - icd10gm2011
      o **OID** = 1.2.276.0.76.5.388
      o **Labels** = \{icd10gm2011\}
Within the current design, Terminology Server is being served externally as a web resource to be invoked by its consumers through RESTful interaction. However, it is probable and possible to deploy the Terminology Server on EHR source sites if any specific requirement arises.

**REST Services of the Terminology Server**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Method</th>
<th>Consumes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>/terminologyserver</td>
<td>HTTP GET</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Retrieve a concept (term), identified with the uri:{uri}**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Method</th>
<th>Consumes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>/terminologyserver/uri?uri={uri}</td>
<td>HTTP GET</td>
<td>-</td>
<td>Uri</td>
</tr>
</tbody>
</table>

**Retrieve a concept (term), identified with the code:{code} from a terminology system which is identified with the OID:{oid}**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Method</th>
<th>Consumes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>/terminologyserver/oid/{oid}/{code}</td>
<td>HTTP GET</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Retrieve a concept (term), identified with the code:{code} from a terminology system which is identified with the label:{label}**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Method</th>
<th>Consumes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>/terminologyserver/label/{label}/{code}</td>
<td>HTTP GET</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameters</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Search the keyword on the terminology system identified with the OID:** `{oid}`

<table>
<thead>
<tr>
<th>Endpoint</th>
<th><code>/terminologyserver/search/oid/{oid}?q=keyword</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>Q</td>
</tr>
</tbody>
</table>

**Search the keyword on the terminology system identified with the label:** `{label}`

<table>
<thead>
<tr>
<th>Endpoint</th>
<th><code>/terminologyserver/search/label/{label}?q=keyword</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>Q</td>
</tr>
</tbody>
</table>

**Retrieve the skos:exact match mappings of the term identified with the code:** `{code}` in the source terminology system identified with the OID: `{oid}`, from the target terminology system identified with the `{toid}`

<table>
<thead>
<tr>
<th>Endpoint</th>
<th><code>/terminologyserver/mappings/exact/oid/{oid}/{code}?toid={toid}</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>Toid</td>
</tr>
</tbody>
</table>

**Retrieve the skos:exact match mappings of the term identified with the code:** `{code}` in the source terminology system identified with the label: `{label}`, from the target terminology system identified with the `{tlabel}`

<table>
<thead>
<tr>
<th>Endpoint</th>
<th><code>/terminologyserver/mappings/exact/label/{label}/{code}?tlabel={tlabel}</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>tlabel</td>
</tr>
</tbody>
</table>

**Retrieve the salusc:exactOrNarrow match mappings of the term identified with the code:** `{code}` in the source terminology system identified with the OID: `{oid}`, from the target terminology system identified with the `{toid}`

<table>
<thead>
<tr>
<th>Endpoint</th>
<th><code>/terminologyserver/mappings/exact-or-narrow/oid/{oid}/{code}?toid={toid}</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>toid</td>
</tr>
</tbody>
</table>

\(^2\) salusc: <http://www.salusproject.eu/ontology/core#>
Retrieve the salusc:exactOrNarrow match mappings of the term identified with the code:{code} in the source terminology system identified with the label:{label}, from the target terminology system identified with the {tlabel}.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/terminologyserver/mappings/exact-or-narrow/label/{label}/{code}?tlabel={tlabel}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>tlabel</td>
</tr>
</tbody>
</table>

Retrieve the salusc:exactOrBroad3 match mappings of the term identified with the code:{code} in the source terminology system identified with the OID:{oid}, from the target terminology system identified with the {toid}.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/terminologyserver/mappings/exact-or-broad/oid/{oid}/{code}?toid={toid}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>toid</td>
</tr>
</tbody>
</table>

Retrieve the salusc:exactOrBroad match mappings of the term identified with the code:{code} in the source terminology system identified with the label:{label}, from the target terminology system identified with the {tlabel}.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/terminologyserver/mappings/exact-or-broad/label/{label}/{code}?tlabel={tlabel}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>application/n-triple, application/rdf+json, application/rdf+xml, texts/n3, text/turtle, application/rdf+cml-abbrev</td>
</tr>
<tr>
<td>Parameters</td>
<td>tlabel</td>
</tr>
</tbody>
</table>

In the first development cycle the integration of Terminology server with the following SALUS components have been finalized:

- Case Series Characterization Tool
- ADE Notification Tool
- ICSR Reporting Tool

The details of these integrations can be found in the following subsections.

As the development of other Safety Analysis Tools including Patient History Tool (PHT), Temporal Association Screening (TAS) Tool, Temporal Pattern Characterization (TPC) Tool, Post Market Safety Study Tool (PMSST); integration of these tools with Terminology Server will be finalized in the next iteration phase, and will be reported in D6.3.2.

---

3 salusc: <http://www.salusproject.eu/ontology/core#>
2.1.1 Case Series Characterization Tool

Case Series Characterization Tool (CSCT) benefits from the Terminology Server to provide a type-ahead search functionality. Type-ahead search functionality is used by safety analysts while creating new eligibility criteria or risk factors.

Thanks to the type-ahead search functionality, safety analysts see relevant terms such as medication, condition, etc. for a given query keyword even though s/he types a part of the intended term that s/he wants to use. CSCT ranks the results according to the similarity of the results thanks to the score information of each result returned by the Terminology Server.

CSCT provides type-ahead search for the terms to be obtained from different terminology systems. For instance, while medication concepts are obtained from the ATC terminology system, condition/problem terms are obtained from the MedDRA terminology system.

When the search results are presented, apart from the labels of the resultant terms, levels of the terms them within the terminology system, such as HLGT, HLT and PT levels from the ATC terminology system, are also presented to the safety analyst.

In the Figure 3, the interaction of the CSCT with the Terminology Server is depicted.

As presented in the Figure 3, CSCT requests similar terms from a particular terminology system for a given query keyword. Below, two curl representation of queries issued by the CSCT to Terminology Server are exemplified:

  This command searches for the condition terms starting with the “myoc” such as myocardial infarction in the MedDRA terminology system.

  This command searches for the medication terms starting with the “nife” such as nifedipine in the ATC terminology system.

2.1.2 ADE Notification Tool

One function of the ADE Notification Tool (ANT) is to check for known ADEs, especially to be able to distinguish between events of that kind and currently unknown ADEs. Therefore, the ANT is connected to some databases containing data on all currently known ADEs. These ADEs are
represented in MedDRA codes. When a diagnosis is checked by the ANT for the presence of a possible known ADE, there is a need for a translation from the source code system of the diagnosis to the MedDRA code system, because the diagnoses are coded in other code systems (e.g. in ICD-9).

To realize this translation ANT will integrate with the terminology server in the following way:

For each diagnosis the terminology server will be queried for the exact matches. The terminology server URL to execute such a query will be as the following one where oid represents the source terminology, code represents the diagnosis for which the exact mappings will be retrieved and toid represents the target terminology system i.e. MedDRA in this case.

http://95.9.71.171:8080/terminologyserver/mappings/exact/oid/{oid}/{code}?toid=2.16.840.1.113883.6.163

If there is no exact mappings for a particular diagnosis, ANT will query the terminology server for exactOrBroad mappings. In this case, the endpoint of the terminology server to be queried would be as follows:

http://95.9.71.171:8080/terminologyserver/mappings/exact-or-broad/oid/{oid}/{code}?toid=2.16.840.1.113883.6.163 &level=PT

This query would obtain close correspondents from the MedDRA terminology system for a particular diagnosis in PT level even if the concepts do not match exactly.

2.1.3 ICSR Reporting Tool

The objective of integrating IRT with Terminology server is to map medical terms in the patient summary (which is provided by SIL-DS and used to prepopulate the ICSR form) with MedDRA terms. MedDRA is indeed the expected terminology in ICSR forms, especially E2B. The method used is to invoke Terminology server to retrieve MedDRA codes for a given non-MedDRA code in the patient summary (ICD, Snomed-CT, etc.). The MedDRA terms provided by Terminology server are used to build an enriched patient summary (see example below), which is then used to prepopulate the ICSR form.

Listing 2 - Extract of patient summary before (left) and after (right) enrichment with MedDRA terms provided by Terminology server.
IRT follows an approach similar to the ANT. In the first step, it queries the exact matches for a particular code. If there is no such mappings, it queries the for the exactOrBroad mappings for that code.

2.2 Integration with Security & Privacy Services

Integration of major SALUS components is handled through RESTful services. Although there are in-machine interactions between SALUS components, there are remote interactions which deeply require security and privacy. That is, while users of the Safety Analysis Tools from the Research Zone accessing the services of the Care Zone and retrieving data, the interactions should be authenticated, secured and audited according to the security concerns of SALUS design. Moreover, privacy concerns about the patient data should be handled before data leaves the Care Zone with the use of deidentification and pseudonymization services. It is important to note that in-machine RESTful interactions of different SALUS components will also be secured.

The subcomponents of salus.security-privacy are there to meet the requirements shortly listed in the previous paragraph. These subcomponents and their objectives are listed with their artefact names as follows:

- **arr**: Stands for Audit Record Repository. As described in the SALUS D3.4.1 Conceptual Design of the SALUS Architecture, SALUS Audit Record Repository is going to implement the IHE ATNA “Audit Record Repository Role”. For this, the open source implementation from Open Health Tools[^4] -- openATNA[^5] -- has been imported since it already provides the repository implementation along with GUIs to navigate the audits in the repository.

- **audit-manager**: In order to interact with the Audit Record Repository, according to the IHE ATNA profile, each auditor should implement the “Secure Node Actor” with ‘ITI-20: Record Audit Event” transaction of the IHE IT Infrastructure Technical Framework[^6]. audit-manager handles this transaction by acting as the passage to interact with the Audit Record Repository. Within the audit-manager, there is an API to be consumed by the auditor components.

- **authentication**: This is the user authentication module for username/password based authentication of the SALUS users. It provides a simple API for the consumption of other components (i.e. cde-repository, sat.csct) to use session based authentication controls of the users.

- **deidentification**: This module provides the deidentification and pseudonymization services to be consumed by other components of SALUS. It can be consumed through Java API calls or RESTful calls by its clients. Privacy issues are handled through configurable deidentification and pseudonymization methods within this module and the Pommerening approach has been partly adopted in which the clients of deidentification module do not receive the deidentified messages back. The messages are sent to their final destinations directly by this module.

- **key-manager**: Secure RESTful communication between different SALUS components are handled through HTTPS[^7]. Remaining network based interactions such as the one between audit-manager and arr, or sending out e-mails for ICSRs are secured with the keys residing in this module.

2.2.1 Integration with Authentication Services

Authentication services are not provided through REST within SALUS. The components who need user authentication makes Java calls to the API of the authentication module. SALUS authentication is managed through session cookies in the web based interaction. Therefore, this module gives session based methods to its consumers. The manager class is the AuthenticationManager which opens up the following Java methods:

[^6]: IT Infrastructure Technical Framework 10 Volume 2a (ITI TF-2a), Section 3.20
Description: This method provides the username/password based user authentication and returns a Session object if the authentication is successful.

Signature: Session login(String username, String password, boolean staySignedIn) throws AuthenticationException

Description: This method handles the sign out operation by removing the session records from the database for that user.

Signature: boolean logoutUserFromSessionID(String sessionID) throws AuthenticationException

Description: The web based components of SALUS which are consuming the authentication services makes a call to this method at the beginning of every transaction that they have with their user in order to check whether the session of the user is valid or not.

Signature: User getUserFromSession(String sessionID) throws AuthenticationException

Following SALUS components are consuming the services provided by the authentication module:

- CDE Repository
- SAT Module (Safety Analysis Tools)
  - Case Series Characterization Tool (CSCT)
  - Patient History Tool (PHT)
  - Temporal Association Screening (TAS) Tool
  - Temporal Pattern Characterization (TPC) Tool
  - Post Market Safety Study Tool (PMSST)

In the first development cycle integration of authentication services with CDE Repository and CSCT is finalized. After the finalization of PHT, PMSST, TAS, TPC tools in the next development cycle, the integration of authentication services will also be finalized.

2.2.2 Integration with Deidentification – Pseudonymization Services

The artefact name for this component is “deidentification”, however; it also provides the pseudonymization related methods to be applied based on the configuration of the requestor. According to the SALUS requirements and conceptual design of SALUS architecture, deidentification component serves three different component within different use cases as in the following:

1. E2B deidentification while sending out ICSRs. The ICSRs are identified based on a configuration to eliminate patient identifying data. ICSR Reporting Tool sends the ICSR to deidentification module which sends the e-mail in an encrypted way to its receiver(s).
2. Patient summary deidentification while presenting the patient summaries to the users of the Research Zone. Patient History Tool (PHT) requests data from the Semantic Interoperability Layer Data Service and then this summary passes through the deidentification module to reach its final destination.
3. Dataset deidentification for the Post Market Safety Study Tool (PMSST). Upon queries made from PMSST to Safety Analysis Query Manager, a dataset is returned as a result. This dataset passes through the deidentification module before sent out to the Research Zone.

Deidentification module opens up a number of RESTful services to provide the functionalities listed above. These services can be listed as follows:

- Deidentify an ICSR based on the Configuration, encrypt and send it as an e-mail to the recipients
indicated inside the ICSR. This method gets two parameters. The first one is a required parameter -- the ICSR which is serialized according to the XML Schema of Ichicrs8 specification. The second one is the configuration which indicates the fields of ICSR to be identified with which methods. If this file is not sent, a default configuration is used for the deidentification of ICSR. Again, a configuration is dictated by an XML Schema residing in the deidentification module9.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/deidentification/e2b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP POST</td>
</tr>
<tr>
<td>Consumes</td>
<td>application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Produces</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>icsr configuration</td>
</tr>
</tbody>
</table>

Deidentify a patient summary structured according to SALUS CIM according to a Configuration. The patientsummary is a required parameter which should be serialized according to the SALUS CIM configuration is an optional parameter to state which fields of the patient summary should be deidentified/pseudonymized with which method. If no configuration is supplied, then a default configuration is used to deidentify the patient summaries to supply patient privacy.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/deidentification/cim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP POST</td>
</tr>
<tr>
<td>Consumes</td>
<td>application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Produces</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>patientsummary configuration</td>
</tr>
</tbody>
</table>

Deidentify a dataset which occurs upon a query from Post Market Safety Study Tool to Safety Analysis Query Manager. This service exists in a conceptual level to be considered in the second implementation loop of SALUS.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/deidentification/pmsstdata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP POST</td>
</tr>
<tr>
<td>Consumes</td>
<td>application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Produces</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Dataset configuration</td>
</tr>
</tbody>
</table>

### 2.2.2.1 ICSR Reporting Tool

The objective of integrating IRT with with Deidentification–Pseudonymization services is to transmit an anonymized or pseudonymized version of the E2B xml report generated by IRT to the pharmacovigilance authorities (UMC in SALUS test phase). The method used is to invoke Deidentification–Pseudonymization services when submitting an E2B xml file. In agreement with the Pommerening approach, the Deidentification–Pseudonymization services are managing the electronic transmission of the report to the pharmacovigilance center. The e-mail address of the receiver of the report is indicated in the xml report (<receiveremailaddress>).

IRT uses the e2b endpoint of the deidentification services. It just sends the complete ICSR and the report is deidentified according to the default configurations including operations such as shifting dates, substituting identifiers, anonymizing conditions, medications, etc.

The integration of ICSR Reporting Tool with De-Identification Module is finalized in this development cycle.

---

8 [http://estri.ich.org/e2br3/index.htm](http://estri.ich.org/e2br3/index.htm)

9 [deidentification-configuration.xsd](file://deidentification-configuration.xsd)
2.2.2.2 Patient History Tool

Patient History Tool aims to present detailed information about a single patient to the analysts in Research Zone. When the analyst in the Research Zone wants to retrieve more information about a patient, she can request this by the help of Patient History Tool. PHT can present patient summaries by retrieving through

- a unique ICSR message numb
- a unique pair of safetyreportid and safetyreportversion
- a unique pseudonym (PSN)

The request of PHT first goes to deidentification module in the Care Zone to retrieve the patient identifier from the given parameter(s) which are described above. Then, Semantic Interoperability Data Service is invoked to retrieve the patient summary and deidentification module again consumed to deidentify and pseudonymize the privacy related fields of the patient summary. Lastly, deidentification module returns the patient summary to the PHT.

The implementation of the PHT tool has not yet been finalized. In the next development cycle upon the finalization of the tool, the integration with De-Identification Module will be finalized immediately as the required interfaces are already implemented.

2.2.2.3 Post Market Safety Study Tool

Post Market Safety Study Tool is the Safety Analysis Tool which resembles to Case Safety Characterization Tool, but with different eligibility criteria and result dataset definitions. However, it also makes its request to the Safety Analysis Query Manager. It sends a query to SAQM to select patients based on an eligibility query. A dataset is prepared by SAQM to be sent back to PHT. At this stage of the execution, the dataset passes through deidentification module. Hence, SAQM sends the dataset to deidentification and then deidentification sends the deidentified and pseudonymized dataset to PHT.

The implementation of the PMSST tool has not yet been finalized. In the next development cycle upon the finalization of the tool, the integration with De-Identification Module will be finalized immediately as the required interfaces are already implemented.

2.2.3 Integration with Audit Services

All interactions between the Care Zone and Research Zone must be audited according to the security and privacy architecture of SALUS. SALUS ships with an Audit Record Repository whose underlying system is OpenATNA. Within the IHE ATNA profile, in order to interact with the repository, the secure nodes (the components who sends audits) should interact with the repository according to a protocol and message specification. To ease the job of SALUS components, audit-manager has been implemented to interact with the Audit Record Repository. audit-manager provides a Java API for all interactions between Care Zone and Research Zone.

Methods of the Audit interface are as follows:

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To be invoked from the De-Identification service in Care Zone before encrypting the ICSR to be sent to the Research Zone. This method gets serialized ICSR content (participantObjectDetailValue) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.</td>
</tr>
</tbody>
</table>
### Signature

```java
void auditIRTDeidentifiedE2B(byte[] patricipantObjectDetailValue,
                               String networkAccessPointID_sender,
                               String networkAccessPointID_receiver);
```

### Description

To be invoked from IRT in Care Zone to audit the ICSR before sending it to AIFA. AIFA report will not be de-identified. This method gets serialized ICSR content (patricipantObjectDetailValue) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditIRT_AIFA(byte[] patricipantObjectDetailValue,
                    String networkAccessPointID_sender,
                    String networkAccessPointID_receiver);
```

### Description

To be invoked by SAQM in Care Zone to audit any request-response pairs made from Research Zone. This method gets serialized request and response message contents (patricipantObjectDetailValue_request, patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditSAQMService(byte[] patricipantObjectDetailValue_request,
                       byte[] patricipantObjectDetailValue_response,
                       String networkAccessPointID_sender,
                       String networkAccessPointID_receiver);
```

### Description

To be invoked by CSCT in Research Zone to audit any request-response pairs made to Care Zone (to SAQM). This method gets serialized request and response message contents (patricipantObjectDetailValue_request, patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditCSCT(byte[] patricipantObjectDetailValue_request,
               byte[] patricipantObjectDetailValue_response,
               String networkAccessPointID_sender,
               String networkAccessPointID_receiver);
```

### Description

To be invoked by TFC tool in Research Zone to audit any request-response pairs made to Care Zone (to SAQM). This method gets serialized request and response message contents (patricipantObjectDetailValue_request, patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.
### Signature

```java
void auditTPC(byte[] patricipantObjectDetailValue_request,
              byte[] patricipantObjectDetailValue_response,
              String networkAccessPointID_sender,
              String networkAccessPointID_receiver);
```

### Description

To be invoked by TAS tool in Research Zone to audit any request-response pairs made to Care Zone (to SAQM). This method gets serialized request and response message contents (patricipantObjectDetailValue_request, patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditTAS(byte[] patricipantObjectDetailValue_request,
              byte[] patricipantObjectDetailValue_response,
              String networkAccessPointID_sender,
              String networkAccessPointID_receiver);
```

### Description

To be invoked by PHT in Research Zone to audit any patient summary request-response pair made to Care Zone (SAQM). Please note that the response will be received from the De-identification service due to the Pommerenning approach. This method gets serialized request and response message contents (patricipantObjectDetailValue_request, patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditPHT(byte[] patricipantObjectDetailValue_request,
              byte[] patricipantObjectDetailValue_response,
              String networkAccessPointID_sender,
              String networkAccessPointID_receiver);
```

### Description

To be invoked from the De-identification service in Care Zone before sending the Patient Summary back to the PHT in Research Zone. This method gets serialized patient summary content (patricipantObjectDetailValue) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

### Signature

```java
void auditDeIdentifiedPatientSummary(byte[]
                                      patricipantObjectDetailValue,
                                      String networkAccessPointID_sender,
                                      String networkAccessPointID_receiver);
```

### Description

To be invoked by the Post Market Safety Study Tool in Research Zone to audit any request-response pair made to Care Zone (to SAQM). Please note that the response will be received from the De-identification service due to the Pommerenning approach. This method gets serialized request object (patricipantObjectDetailValue_request) and deidentified patient summary content (patricipantObjectDetailValue_response) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as
parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

**Signature**
```java
void auditPMSST(byte[] participantObjectDetailValue_request, byte[] participantObjectDetailValue_response, String networkAccessPointID_sender, String networkAccessPointID_receiver);
```

**Description**
To be invoked from the De-identification service in Care Zone before sending the PMSST Dataset back to the Post Market Safety Study Tool. This method gets serialized PMSST dataset content (participantObjectDetailValue) and IP addresses of both sender and receiver (networkAccessPointID_sender, networkAccessPointID_receiver) as parameters and sends the related audit message to the Audit Record Repository. IP addresses are optional and can be set to empty String if they are not available.

**Signature**
```java
void auditDeidentifiedPMSSTdataset(byte[] participantObjectDetailValue, String networkAccessPointID_sender, String networkAccessPointID_receiver);
```

These services will be used by the following tools as indicated in the description of the tools.

- SAT Module (Safety Analysis Tools)
  - Case Series Characterization Tool (CSCT)
  - Patient History Tool (PHT)
  - Temporal Association Screening (TAS) Tool
  - Temporal Pattern Characterization (TPC) Tool
  - Post Market Safety Study Tool (PMSST)
- De-Identification Service
- Safety Analysis Query Manager
- ICSR Reporting Tool

The integration will be finalized in the next development cycle.

### 2.3 Integration with Semantic Services

The Semantic Services are responsible for implementing the data provisioning requirements of the tools. They are called semantic services because they work on formal semantic data. The Semantic Services are mainly managers that are responsible for managing the subscription to or querying of data from the connected data sources, performing calculations on the retrieved data, having the result converted to the requested content model and format, and providing the final data to the requesting tools. So in fact these managers are dedicated) workflow engines controlling the data provisioning process on behalf of the SALUS safety analysis tools.

Looking at the current implementation status, there is only one semantic service providing RESTful services to be consumed through the HTTP protocol. So, in the following sections the integration of different components with this single semantic service i.e. the Safety Analysis Query Manager will be explained.
2.3.1 Integration with Safety Analysis Query Manager

The Safety Analysis Query Manager (SAQM) consumes SIL-DS’s service that allows querying the EHR sources with the eligibility criteria provided by the safety analysis tools such as the Case Series Characterization Tool and Temporal Pattern Characterization Tool.

In the case series characterization scenario, this component applies statistical analysis, which was configured by the safety analysts as well, on top of the retrieved patient data. While applying the statistical analysis, it also executes a terminology reasoning which allows grouping of the concept terms according to a target terminology system level configured by the safety analyst through the safety analysis tools as well. For instance, in the LISPA case, although the condition codes are coded with the ICD-9-CM terminology system, the safety analyst may want to see the results in MedDRA codes.

The following services are provided by this component:

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/safetyanalysisquerymanager/population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP POST</td>
</tr>
<tr>
<td>Consumes</td>
<td>application/json</td>
</tr>
<tr>
<td></td>
<td>The JSON serialization of the SAQMCSCRequest object containing the following inner parameters: • Eligibility criteria • Criteria source indicating whether the eligibility criteria are for the foreground or for the background population. • Statistical options to be processed by the Query Result Calculator to apply statistical safety analysis queries on the data sets once they are collected from the EHR sources. • Risk factors to be processed by the Query Result Calculator to apply statistical safety analysis queries on the data sets once they are collected from the EHR sources.</td>
</tr>
<tr>
<td>Produces</td>
<td>application/json</td>
</tr>
<tr>
<td></td>
<td>The JSON serialization of the List&lt;ResultItem&gt; object containing the statistical analysis results for the eligible patient data</td>
</tr>
<tr>
<td>Parameters</td>
<td>-</td>
</tr>
</tbody>
</table>

2.3.1.1 Case Series Characterization Tool

The Case Series Characterization Tool (CSCT) enables the query of data sources for EHR extracts of selected patient population i.e. the foreground population to characterize ADE cases that originate from SALUS EHR data and compares the statistics against a custom background population. It provides a graphical interface to identify the eligibility criteria of the selected patient populations, and also to list the required statistical comparisons between the collected data and background population. It also provides a graphical interface to present the resulting statistics to the safety analyst.

CSCT queries the Safety Analysis Query Manager (SAQM) in a concurrent way such that it initiates a new process for each EHR source. The EHR source dedicated processes are further divided into two separate processes for foreground and background populations. Each of these leaf level processes queries the SAQM with all of the parameters introduced by the safety analyst.
Figure 4 summarizes the integration between the SAQM and CSCT.

![Diagram of SAQM and CSCT integration](image)

**Figure 4 - Integration between the SAQM and CSCT**

### 2.3.1.2 Post Market Safety Study Tool

Post Market Safety Study Tool (PMSST) will be one of the consumers of the SAQM. The usage of the SAQM by PMSST is similar to the Case Series Characterization Tool case. That is, PMSST benefits from the SAQM’s service that provides querying the EHR sources with an eligibility criteria; the safety analyst would like to obtain a population of patients given an eligibility criteria. As the PMSST have not been finalized yet, this integration has not been realized yet. However the required interfaces are ready and already used and tested in the integration of CSCT.

### 2.4 Integration with Semantic Interoperability Layer

Semantic Interoperability Layer (SIL) provides a number of components that enables semantic interoperability between clinical research systems. It semantically lifts the raw patient data obtained from the EHR sources in local format such as standards like HL7/ASTM CCD, or proprietary models like ORBIS Data Model. As a result, the patient data is represented in RDF format. To provide interoperability, the RDF representations of local models are transformed into a common model called SALUS Common Information Model (CIM) through reasoning operations.

#### 2.4.1 Integration with Semantic Interoperability Layer Data Service

The Semantic Interoperability Layer Data Services (SIL-DS) is responsible for providing a uniform view on the connected EHR data sources using the SALUS CIM. This service accepts queries expressed using the SALUS CIM. Different components send queries to this module according to their needs. Integration of the SIL-DS with different components will be explained in the following sections. Before that different services provided by the SIL-DS component will be given below.

<table>
<thead>
<tr>
<th>Returns all medical history of a patient queried with an identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Endpoint</strong></td>
</tr>
<tr>
<td><strong>Method</strong></td>
</tr>
<tr>
<td><strong>Consumes</strong></td>
</tr>
<tr>
<td><strong>Produces</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns all medical histories of patients matching with the provided eligibility query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Endpoint</strong></td>
</tr>
<tr>
<td><strong>Method</strong></td>
</tr>
<tr>
<td><strong>Consumes</strong></td>
</tr>
<tr>
<td><strong>Produces</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
</tbody>
</table>

Returns identifiers of patients whose medical history is updated between
the given begin and end dates

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>text/turtle # The list of patient IDs who have updates will be returned. There may be a blank node (or the URL of the request itself) as a root which includes a list of patient identifiers. Example: :salus:includes &lt;PID&gt;.</td>
</tr>
<tr>
<td>Parameters</td>
<td>begin= (REQUIRED) # The begin date after which the updates from the database will be sent. end= (REQUIRED) # The end date before which the updates from the database will be sent.</td>
</tr>
</tbody>
</table>

Returns the information for a specific entity such as condition, medication, etc. between the given begin and end dates for all patients

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>/{entity}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HTTP GET</td>
</tr>
<tr>
<td>Consumes</td>
<td>-</td>
</tr>
<tr>
<td>Produces</td>
<td>text/turtle # All data for the given {entity} (condition, medication, ...) for all patients in the given period will be retrieved from the EHR database.</td>
</tr>
<tr>
<td>Parameters</td>
<td>{entity} as a path parameter begin= (REQUIRED) # The begin date of the period. end= (REQUIRED) # The end date of the period.</td>
</tr>
</tbody>
</table>

2.4.1.1 Safety Analysis Query Manager

Safety Analysis Query Manager (SAQM) consumes a particular SIL-DS service which provides querying the EHR sources with an eligibility criteria.

SAQM is a mediator between the Case Series Characterization Tool (CSCT) and SIL-DS. CSCT sends the eligibility criteria defined by the safety analysts. Based on this eligibility criteria SAQM retrieves the patient data from the EHR source. Once the data is obtained, the statistical analysis methods, which was also defined by the safety analysts and passed by the CSCT, are executed on the data and the results are returned back to the CSCT.

Below a code snipped invoking the RESTful service of the SIL-DS service can be seen.

```java
StringWriter out = new StringWriter();
eligibilityCriteria.toRDF().getModel().write(out, "TURTLE");
String serializedQuery = out.toString();
patientData = SIL.header("Content-Type", "text/turtle; charset=UTF-8").accept("text/turtle").post(String.class, serializedQuery);
```

Since the SIL-DS accepts eligibility criteria in RDF format which compatible with the SALUS CIM, the eligibility criteria passed to the SAQM is first converted into the RDF format and then it is passed as the payload of the POST request being sent to the SIL-DS.

2.4.1.2 ADE Notification Tool

By the use of the SIL DS, the ANT queries the underlying EHR system to receive the data in the SALUS CIM format via a SPARQL endpoint. The first prototype uses the SIL DS to query for a single patient. To retrieve all information from a specific patient, the following query is used:

```sparql
CONSTRUCT {
?s ?p ?o
}"
The subsequence “<PATIENTID>” has to be replaced by the patient ID. Afterwards the ANT Content Formatter converts the query results into the ADE content format that can be processed by the tool components (see section 5.2.1 in D6.1.1 for further details).

The second prototype will use a subscription based mechanism provided by the SIL DS to receive data from the moment it is available in the EHR system.

### 2.4.1.3 ICSR Reporting Tool

The objective of integrating IRT with SIL-DS is to obtain the patient summary needed for prepopulating the ICSR form. The method used is to invoke SIL-DS service based on the PID received from ANT. Once the patient summary is received by IRT from SIL-DS, SPARQL queries are used for extracting data for pre-populating candidate data elements in E2B, and format and value set conversions are performed in case of data format and value sets mismatch.

### 2.4.1.4 OMOP Converter

Although OMOP Converter has not been implemented yet, conceptually this component will consume the SIL-DS to query the patient data inserted in a given date range. Since there will be too much data to fit in the physical memory of the computers, OMOP converter will iteratively query the particular entities of patients such as condition, medication, etc. in a preconfigured period range e.g. monthly. OMOP Converter will populate the OMOP database which would be used by the Temporal Association Screening and Temporal Pattern Characterization use cases with the data to be obtained from the SIL-DS services.

An example RESTful call to the SIL-DS services would be as follows:

```bash
```

The RESTful service call above provides the retrieval of the patient data added into the EHR source in the first month of the 2013. The integration will be realized in the next development cycle.

### 2.4.2 Integration with Content Formatter

Content Formatter is the component converting the queries in SALUC CIM format into the local query model. Vice versa, it converts the medical history of patients obtained from the EHR sources into the SALUS CIM format. Since there is not a need to access Content Formatter’s services remotely, the services provided by this component are exposed through the Java API.
Following services are provided by this component

<table>
<thead>
<tr>
<th>Description</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converts an eligibility criteria represented in the SALUS CIM into to the QEDExt format and returns the QEDExt compatible query.</td>
<td>QUPCIN043100UV01 convertToQEDExt(EligibilityCriteria criteria)</td>
</tr>
<tr>
<td>Wraps a patient id with a QEDExt compatible query and returns it as the function response</td>
<td>QUPCIN043100UV01 convertToQEDExt(String patientId)</td>
</tr>
<tr>
<td>Takes the medical history data of patients in RDF format and translates it into the SALUS CIM compatible RDF.</td>
<td>InputStream convertToCIM(String qedRdfTranslated)</td>
</tr>
</tbody>
</table>

2.4.2.1 Semantic Interoperability Layer Data Service
SIL-DS implementation for the LISPA pilot site needs to delegate the incoming queries to the QEDExt Web Services. To convert these queries into a format which is compatible with the target web services, this module benefits from the functionalities provided by the Content Formatter.

2.5 Integration with Technical Interoperability Layer
The Technical Interoperability Layer (TIL) has several interfaces and can be used in many different ways. Within the following the interfaces are described in detail.
Figure 5: Overview of the Technical Interoperability Layer focusing on interfaces

Figure 5 gives an overview of the Technical Interoperability Layer focusing on the interfaces. Within the project it is planned to use the chain through TIL completely from left to right. A JAVA native interface already described in D5.1.1 section 4.2.2 allows a JAVA application to use the Data Consumer (left block in the figure) through the Workflow Management. The query to be either defined for IHE QEDext or IHE CMext is transported by the Workflow Management and afterwards performed by the Data Consumer. Data Consumer and Data Repository are linked to each other by using the IHE profiles defined in D5.1.1 and D5.2.1. This communication is Web-Service based and defined through WSDL. This allows also none JAVA applications to connect to the system. The Data Repository is again a JAVA application which forwards the query by using the Workflow Management to the EHR Connector. In SALUS pilot this EHR connector is called LISPA connector and described in D5.2.1. It analysis the query defined ether in IHE QEDext or IHE CMext and performs the corresponding SQL-queries on the underlying database. In the pilot scenario the results are fetched from LISPA database and returned as results in IHE QEDext or IHE CMext to the Data Consumer querying for.

As the most important interface for TIL is the Web-Service in between the IHE profiles, the QEDext profile for Data Consumer and Data Repository is described within the following.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 xmlns:tns="urn:ihe:pcc:qed:2007"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:ns2="http://schemas.xmlsoap.org/soap/http"
 xmlns:ns1="urn:hl7-org:v3" name="CareManager_Service"
 targetNamespace="urn:ihe:pcc:qed:2007">
 <wsdl:types>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```
targetNamespace="urn:hl7-org:v3"

elementFormDefault="qualified">
  <xs:import namespace="urn:hl7-org:v3"
    schemaLocation="processable/multicacheschemas/QUPC_IN043100UV01.xsd" />
  <xs:import namespace="urn:hl7-org:v3"
    schemaLocation="processable/multicacheschemas/QUPC_IN043200UV01.xsd" />
  <xs:import namespace="urn:hl7-org:v3"
    schemaLocation="processable/multicacheschemas/MCCI_IN000002UV01.xsd" />
  <xs:import namespace="urn:hl7-org:v3"
    schemaLocation="processable/multicacheschemas/QUQI_IN000003UV01.xsd" />
  <xs:import namespace="urn:hl7-org:v3"
    schemaLocation="processable/multicacheschemas/QUPC_MT040300UV01.xsd" />
</xs:schema>
</wsdl:types>

<wsdl:message name="CareManager_QUPC_IN043100UVResponse">
  <wsdl:part element="ns1:MCCI_IN000002UV01" name="Body">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="CareManager_QUPC_IN043100UV">
  <wsdl:part element="ns1:QUPC_IN043100UV01" name="Body">
  </wsdl:part>
</wsdl:message>

<wsdl:portType name="CareManager_PortType">
  <wsdl:operation name="CareManager_QUPC_IN043100UV">
    <wsdl:input message="tns:CareManager_QUPC_IN043100UV"
      name="CareManager_QUPC_IN043100UV"/>
    <wsdl:output message="tns:CareManager_QUPC_IN043100UVResponse"
      name="CareManager_QUPC_IN043100UVResponse"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="CareManager_ServiceSoapBinding"
type="tns:CareManager_PortType">
  <soap12:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="CareManager_QUPC_IN043100UV">
    <soap12:operation soapAction="urn:hl7-org:v3:QUPC_IN043100UV"
      style="document" />
    <wsdl:input name="CareManager_QUPC_IN043100UV">
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output name="CareManager_QUPC_IN043100UVResponse">
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="CareManager_Service">
  <wsdl:port binding="tns:CareManager_ServiceSoapBinding"
    name="CareManager_PortSoap12">
    <!-- location is automatically set when deployed on tomee -->
    <!-- however this here needed for integration tests with an
    embedded server -->
    <soap12:address location="http://localhost:8080/qedext-ws/services/cmWebService12" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
This WSDL file describes the Web-Service of the Data Consumer and is used whenever a Data Repository reacts on an IHE QEDext query. The result set is sent to this Web-Service. The WSDL description of the IHE CMext Web-Service is currently unfinished and will be presented in the next version of the deliverable.

Within the following the WSDL description of the Data Repository is described. It will be used whenever a query using IHE QEDext is send to the Data Repository querying for data. Again the WSDL description for IHE CMext is currently unfinished and will be described in the next version of the deliverable.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
    xmlns:tns="urn:ihe:pcc:qed:2007"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/
    xmlns:ns2="http://schemas.xmlsoap.org/soap/http"
    xmlns:ns1="urn:hl7-org:v3"
    targetNamespace="urn:ihe:pcc:qed:2007">
    <wsdl:types>
        <xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            targetNamespace="urn:hl7-org:v3"
            elementFormDefault="qualified">
            <xs:import namespace="urn:hl7-org:v3"
                schemaLocation="processable/multicacheschemas/QUPC_IN043100UV01.xsd" />
            <xs:import namespace="urn:hl7-org:v3"
                schemaLocation="processable/multicacheschemas/QUPC_IN043200UV01.xsd" />
            <xs:import namespace="urn:hl7-org:v3"
                schemaLocation="processable/multicacheschemas/MCCI_IN000002UV01.xsd" />
            <xs:import namespace="urn:hl7-org:v3"
                schemaLocation="processable/multicacheschemas/QUQI_IN000003UV01.xsd" />
            <xs:import namespace="urn:hl7-org:v3"
                schemaLocation="processable/multicacheschemas/QUPC_MT040300UV01.xsd" />
        </xs:schema>
    </wsdl:types>

    <wsdl:message name="ClinicalDataSource_QUQI_IN000003UV01_Cancel">
        <wsdl:part element="ns1:QUQI_IN000003UV01" name="Body" />
    </wsdl:message>

    <wsdl:message name="ClinicalDataSource_QUPC_IN043200UVResponse">
        <wsdl:part element="ns1:MCCI_IN000002UV01" name="Body" />
    </wsdl:message>

    <wsdl:message name="ClinicalDataSource_QUPC_IN043100UVResponse">
        <wsdl:part element="ns1:QUPC_IN043200UV01" name="Body" />
    </wsdl:message>

    <wsdl:message name="ClinicalDataSource_QUPC_IN043200UV">
        <wsdl:part element="ns1:QUPC_IN043200UV01" name="Body" />
    </wsdl:message>

    <wsdl:message name="ClinicalDataSource_QUQI_IN000003UV01_CancelResponse">
        <wsdl:part element="ns1:QUPC_IN043200UV01" name="Body" />
    </wsdl:message>
</wsdl:definitions>
```
<wsdl:message name="ClinicalDataSource_QUQI_IN000003UV01_ContinueResponse">
  <wsdl:part element="ns1:QUQI_IN000003UV01" name="Body" />
</wsdl:message>

<wsdl:message name="ClinicalDataSource_QUQI_IN000003UV01_Continue">
  <wsdl:part element="ns1:QUQI_IN000003UV01" name="Body" />
</wsdl:message>

<wsdl:message name="ClinicalDataSource_QUPC_IN043100UV">
  <wsdl:part element="ns1:QUPC_IN043100UV01" name="Body" />
</wsdl:message>

<wsdl:portType name="ClinicalDataSource_PortType">
  <wsdl:operation name="ClinicalDataSource_QUPC_IN043100UV">
    <wsdl:input message="tns:ClinicalDataSource_QUPC_IN043100UV"
                name="ClinicalDataSource_QUPC_IN043100UV">
    </wsdl:input>
    <wsdl:output message="tns:ClinicalDataSource_QUPC_IN043100UVResponse"
                 name="ClinicalDataSource_QUPC_IN043100UVResponse">
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClinicalDataSource_QUPC_IN043200UV">
    <wsdl:input message="tns:ClinicalDataSource_QUPC_IN043200UV"
                name="ClinicalDataSource_QUPC_IN043200UV">
    </wsdl:input>
    <wsdl:output message="tns:ClinicalDataSource_QUPC_IN043200UVResponse"
                 name="ClinicalDataSource_QUPC_IN043200UVResponse">
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClinicalDataSource_QUQI_IN000003UV01_Continue">
    <wsdl:input message="tns:ClinicalDataSource_QUQI_IN000003UV01_Continue"
                name="ClinicalDataSource_QUQI_IN000003UV01_Continue">
    </wsdl:input>
    <wsdl:output message="tns:ClinicalDataSource_QUQI_IN000003UV01_ContinueResponse"
                 name="ClinicalDataSource_QUQI_IN000003UV01_ContinueResponse">
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClinicalDataSource_QUQI_IN000003UV01_Cancel">
    <wsdl:input message="tns:ClinicalDataSource_QUQI_IN000003UV01_Cancel"
                name="ClinicalDataSource_QUQI_IN000003UV01_Cancel">
    </wsdl:input>
    <wsdl:output message="tns:ClinicalDataSource_QUQI_IN000003UV01_CancelResponse"
                 name="ClinicalDataSource_QUQI_IN000003UV01_CancelResponse">
    </wsdl:output>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="ClinicalDataSource_ServiceSoapBinding" type="tns:ClinicalDataSource_PortType">
  <soap12:binding style="document"
               transport="http://schemas.xmlsoap.org/soap/http" />
</wsdl:binding>
<wsdl:operation name="ClinicalDataSource_QUPC_IN043100UV">
  <soap12:operation soapAction="urn:hl7-org:v3:QUPC_IN043100UV" style="document" />
  <wsdl:input name="ClinicalDataSource_QUPC_IN043100UV">
    <soap12:body use="literal" />
  </wsdl:input>
  <wsdl:output name="ClinicalDataSource_QUPC_IN043100UVResponse">
    <soap12:body use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="ClinicalDataSource_QUPC_IN043200UV">
  <soap12:operation soapAction="urn:hl7-org:v3:QUPC_IN043200UV" style="document" />
  <wsdl:input name="ClinicalDataSource_QUPC_IN043200UV">
    <soap12:body use="literal" />
  </wsdl:input>
  <wsdl:output name="ClinicalDataSource_QUPC_IN043200UVResponse">
    <soap12:body use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="ClinicalDataSource_QUQI_IN000003UV01_Cancel">
  <soap12:operation soapAction="urn:hl7-org:v3:QUQI_IN000003UV01_Cancel" style="document" />
  <wsdl:input name="ClinicalDataSource_QUQI_IN000003UV01_Cancel">
    <soap12:body use="literal" />
  </wsdl:input>
  <wsdl:output name="ClinicalDataSource_QUQI_IN000003UV01_CancelResponse">
    <soap12:body use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="ClinicalDataSource_QUQI_IN000003UV01_Continue">
  <soap12:operation soapAction="urn:hl7-org:v3:QUQI_IN000003UV01_Continue" style="document" />
  <wsdl:input name="ClinicalDataSource_QUQI_IN000003UV01_Continue">
    <soap12:body use="literal" />
  </wsdl:input>
  <wsdl:output name="ClinicalDataSource_QUQI_IN000003UV01_ContinueResponse">
    <soap12:body use="literal" />
  </wsdl:output>
</wsdl:operation>

2.5.1 Semantic Interoperability Layer Data Service
Semantic Interoperability Layer Data Service (SIL-DS) component of SALUS project provides different querying functionalities such as querying all medical history of a patient with a given pseudonymized id, querying particular entities of patients updated within a certain date range, etc. on
top of the underlying EHR sources. The SIL-DS implementation for LISPA side uses the Technical Interoperability Layer (TIL) services to delegate the incoming queries to the underlying EHR source.

Various SIL-DS implementations in SALUS project implements a unified interface declaring the service signatures and format and type of the returned objects from the provided services. Based on this interface SIL-DS implementations accept parameters compatible with the SALUS CIM e.g. an eligibility query. On the other hand, TIL provides Web Services accepting parameters compatible with the QEDExt standard which is a slightly extended version of QED standard. This extension is described in the Deliverable 5.2.1 Query Based Interoperability Profiles and Open Source Toolsets in detail.

Since TIL services accepts parameters in QEDExt format, SIL-DS implementation for the LISPA side first transforms the parameters of incoming queries into the QEDExt format which can be processed by the TIL.

Below a little code snippet depicting the TIL Web Service call for querying the EHR source with an eligibility query given is given:

```java
EligibilityCriteria eligibilityCriteria=... // a service parameter

ClinicalDataSourceService service = new ClinicalDataSourceService(new URL(WSDL_URL),ClinicalDataSourceService.SERVICE);
ClinicalDataSourcePortType port = service.getClinicalDataSourcePortSoap12();
mapLocalTerminologyCodes(eligibilityCriteria);
QUPCIN043100UV01 query = QEDExtFormatter.convertToQEDExt(eligibilityCriteria);
QUPCIN043200UV01 response = port.clinicalDataSourceQUPCIN043100UV(query);
InputStream patientData = QEDExtFormatter.convertToCIM(patientDataFile);
writePatientDataToResponse(patientData);
```

According to the code snippet above, in the first step a client for the TIL Web Service is created using the predefined WSDL URL of the Web Services. The WSDL has already been explained in the Section 2.5.

In the next step, if the concepts passed in the eligibility criteria are coded with a different terminology system than the local terminology system of the EHR source, the codes included in the eligibility criteria are replaced with the correspondent codes contained in the local terminology system. In case of the LISPA pilot site, the target terminology system is ICD-9-CM.

In the third step, the eligibility criteria object which is compatible with the SALUS CIM is transformed into the QUPCIN043100UV01 type which is a type defined in the QED standard.

In the fourth step, the QEDExt compatible query instance is passed to the Web Service client to query the eligible patients from the EHR source. The response is received in QUPCIN043200UV01 type which is a type defined in the QED standard as well.

In the last step, the response in QEDExt format is converted into the SALUS CIM format and the converted data is written to the response.

All of the querying functionalities provided by the LISPA implementation of the SIL-DS component follows the same steps in terms of the integration with the TIL.
### 2.6 ANT – IRT integration

The objective of integrating IRT with ANT is to enable ANT to trigger the creation of a new ICSR and to transmit several parameters from ANT to IRT: (i) the PID which is needed to call SIL-DS and (ii) additional data (when available) needed for prepopulation. For this integration, IRT has been exposed as a service to be called by ANT when an ADE must be reported: ANT invokes IRT via URL and passes the required parameter as "parameters" in the URL.

When ANT calls IRT as a service, the following list of parameters, which depends of the kind of ADE to be reported and the rule that has been used to detect it, is transmitted:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient_ID</td>
<td>String</td>
<td>Necessary to call SIL-DS</td>
</tr>
<tr>
<td>ADE_Detection_Type</td>
<td>String</td>
<td>0: manual ADE detection, 1: detection based on known ADE DB, 2: detection based on lab test results (TUD), 3: detection based on LISPA rules</td>
</tr>
<tr>
<td>Reporting</td>
<td>String</td>
<td>0: ADE to be reported later, 1: ADE to be reported now</td>
</tr>
<tr>
<td>ADE_Source_Code</td>
<td>String</td>
<td>ADE source code</td>
</tr>
<tr>
<td>ADE_Source_Coding_System</td>
<td>String</td>
<td>ADE source coding system</td>
</tr>
<tr>
<td>ADE_MedDRA_Code</td>
<td>String</td>
<td>ADE MedDRA code</td>
</tr>
<tr>
<td>ADE_Start_Date</td>
<td>String</td>
<td>Date of the beginning of ADE occurrence</td>
</tr>
<tr>
<td>ADE_End_Date</td>
<td>String</td>
<td>Date of the end of ADE occurrence</td>
</tr>
<tr>
<td>Drug_Code</td>
<td>String</td>
<td>Code of the drug</td>
</tr>
<tr>
<td>Drug_Coding_System</td>
<td>String</td>
<td>Drug coding system</td>
</tr>
<tr>
<td>Drug_Start_Date</td>
<td>String</td>
<td>Date of the beginning of drug administration</td>
</tr>
<tr>
<td>Drug_End_Date</td>
<td>String</td>
<td>Date of the end of drug administration</td>
</tr>
<tr>
<td>Justification_Trace</td>
<td>String</td>
<td>Reason for reporting the ADE</td>
</tr>
</tbody>
</table>

Depending on the value of parameter 2 “ADE_Detection_Type”, the parameters transmitted from ANT to IRT are different:

- Case #0: Only <1, 2> are transmitted
- Case #1 #2 #3: <1-13> are transmitted (although some parameters may have empty values when data is not available)

Below is an example request to invoke IRT via ANT, for the Patient_ID = 883003 and other parameters.

```
```

"http://localhost:8080/SalusProj/form" will be changed based on deployment configurations.
3 CONCLUSION & FUTURE WORK

In this deliverable we have presented the integration activities carried out in the first development cycle of SALUS Project. Several tools are currently under development and will be finalized in the second development cycle. As has been described in the sections above, several additional integration activities are planned to be carried out in the second development cycle related with these new components. These can be summarized as follows:

- Integration of Audit Record Services with:
  - Case Series Characterization Tool (CSCT)
  - Patient History Tool (PHT)
  - Temporal Association Screening (TAS) Tool
  - Temporal Pattern Characterization (TPC) Tool
  - Post Market Safety Study Tool (PMSST)
  - Safety Analysis Query Manager
  - ICSR Reporting Tool
  - De-Identification Service

- Integration of Patient History Tool (PHT)
  - Safety Analysis Query Manager
  - De-Identification Service
  - Semantic Interoperability Layer

- Integration of Temporal Association Screening (TAS) Tool
  - OMOP Converter
  - Safety Analysis Subscription Manager
  - Semantic Interoperability Layer

- Integration of Temporal Pattern Characterization (TPC) Tool
  - OMOP Converter
  - Safety Analysis Subscription Manager
  - Semantic Interoperability Layer

- Integration of Post Market Safety Study Tool (PMSST)
  - Safety Analysis Query Manager
  - De-Identification Service
  - Semantic Interoperability Layer

- Integration of TIL with
  - Safety Analysis Subscription Manager

- Integration of SIL with
  - OMOP converter

- Integration of ANT with
  - Safety Analysis Subscription Manager

In addition to these, the ADE Notification Tool and the ICSR Reporting Tools that will be used by the general practitioners and the physicians will be integrated with the existing tools in LISPA and TUD. These will be reported in the second release of this deliverable, namely D6.3.2 at Month 26.