Integrative Cancer Research
INTEGRATE
Through Innovative Biomedical Infrastructures

SEVENTH FRAMEWORK PROGRAMME

ICT-2010-270253

INTEGRATE

Driving excellence in Integrative Cancer Research through Innovative Biomedical Infrastructures

STREP
Contract Nr: 270253

Deliverable: D2.5 Integration Guidelines

Due date of deliverable: 31-07-2012
Actual submission date: 21-01-2013

Start date of Project: 01 February 2011 Duration: 36 months

Responsible WP: Philips

Revision: <outline, draft, proposed, **accepted**>

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Service	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Antonio Rico	UPM	arico@infomed.dia.fi.upm.es
Brecht Claerhout	Custodix	Brecht.claerhout@custodix.com
David Pérez	UPM	dperez@infomed.dia.fi.upm.es
Georgios Manikis	FORTH	gmanikis@ics.forth.gr
Ioannis Karatzanis	FORTH	karatzan@ics.forth.gr
Jasper van Leeuwen	Philips	Jasper.van.Leeuwen@philips.com
Kristof De Schepper	Custodix	kristof.deschepper@custodix.com
Raúl Alonso	UPM	ralonso@infomed.dia.fi.upm.es
Sergio Paraíso	UPM	sparaiso@infomed.dia.fi.upm.es

0.2 Documents history

Document version #	Date	Change
V0.1	29/10/2012	Definition of ToC
V0.2	11/1/2013	Updated version (send to project internal reviewers)
V0.3	15/1/2013	Signed off version (for approval to PMT members)
V1.0	21/1/2013	Approved Version to be submitted to EU

0.3 Document data

Keywords	Guidelines, services, integration
Editor Address data	Name: Kristof De Schepper Partner: Custodix Address: Korterijksesteenweg 214 bus 3 B-9830 Sint-Martens-Latem Phone: +32 (0) 9 210 78 90 Fax: +32 (0) 9 210 78 90 E-mail: kristof.deschepper@custodix.com
Delivery date	

0.4 Distribution list

Date	Issue	E-mailer
15/01/2013	V0.3	Fp7-integrate@listas.fi.upm.es
21/01/2013	V1.0	Robert.BEGIER@ec.europa.eu CNECT-ICT-270253@ec.europa.eu Kinga.VICZIAN@ec.europa.eu

Table of Contents

1 Introduction

In this deliverable, integration guidelines for the INTEGRATE platform are described. The integration guidelines cover the integration of new data sources, the integration of new components into the platform and the integration of new models and analytical tools.

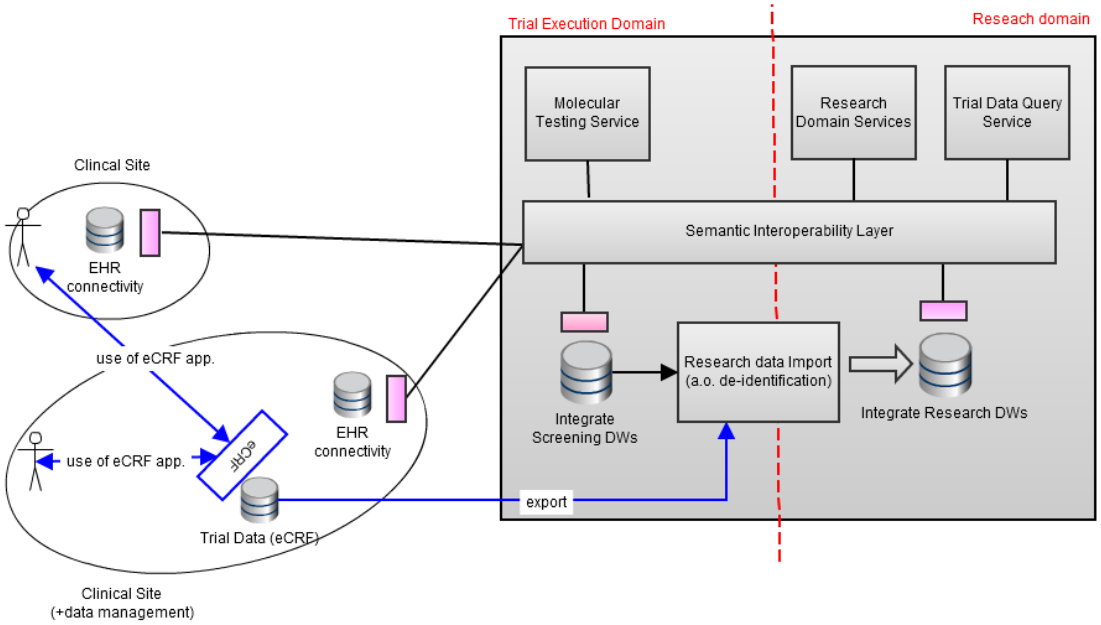


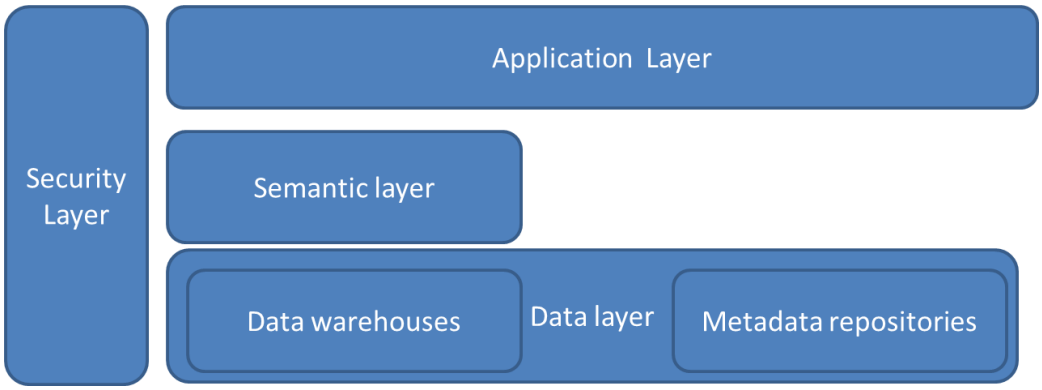
Figure 1 - 10000 feet view overall architecture

Figure 1 shows a high level overview of the initial architecture as described in deliverable 2.4. Section 2.3 describes the semantic approach taken in the INTEGRATE project and provides guidelines of how to incorporate a new data source into the platform by describing the steps which are necessary to extract, translate and load the data residing in the new data source into one of the INTEGRATE data warehouses (either an existing data warehouse or a new instance). This approach provides a low barrier method of inclusion of a new data source into the platform. An alternative approach would be to complement the data source with an interface conforming to the INTEGRATE data warehouse interface specifications, as touched upon in Section 2.1. Section 2.1 also describes the integration of new application layer services. Finally, the INTEGRATE platform provides analytic tools and models. Section 2.3 describes how new tools and models can be added to the INTEGRATE platform.

2 Technical approach

2.1 Application Layer Services

The INTEGRATE platform takes a service oriented architecture approach. The focus of the software development effort is on developing loosely coupled software components, characterized by their (well-defined) functionality. The functionality of the components is exposed by means of a well-defined, explicit interface. Technically, the consortium has chosen to implement the architecture using web services (WSDL/SOAP), which ensures that functionality can be provisioned independent of platforms and programming languages. The approach taken ensures the maximum potential of re-use of developed components.



The INTEGRATE platform is designed using a multi-layered architecture, with responsibilities assigned to the various layers. In INTEGRATE, we have the data layer at the bottom. The data layer contains the various data warehouses (such as the screening data warehouses, research data warehouses and other clinical warehouses) and the metadata repositories (such as the trial metadata repositories and the model repositories). The clinical data warehouses all expose a standardized query interface, and queries are expressed using the INTEGRATE core dataset.
On top of the data layer lays the semantic layer. This layer provides reasoning capabilities for querying the data warehouses and provides access to (the concepts in) the core dataset.
Subsequently, we have the Application layer. This layer houses a variety of application services. Where possible, INTEGRATE promotes the approach of providing re-usable services in the application layer as well. Typically, the applications in the application layer contain internal layering (for instance the introduction of a view layer, etc), but the overall INTEGRATE platform does not require this. The application layer provides a number of default services such as the molecular testing services, Central pathology review services, and Analytical tool services.

On an orthogonal axis the security layer can be found. This layer is used by all other layers. The security layer provisions identity, authentication and de-identification services (see section 2.2 for more details).

In order to provide new functionality in the application layer, the developed software should comply with the following guidelines:

- The software should comply with the authentication and authorization rules in the INTEGRATE platform (see section 2.2 for more details).
- In case the software provides a service interface, it should use be provided using WS-I Basic Profile.
- The deployment of the component should comply with the deployment rules as specified in deliverable 6.2.

2.2 Integration with Security Services (Vertical Layer)

2.2.1 Introduction

The different services constituting the INTEGRATE security layer have been explained in Deliverable 2.4. As mentioned there, the INTEGRATE security solution consists of re-usable modular components that respectively deal with authentication (authN), authorisation (authZ), audit and privacy enhancing (i.e. services oriented specifically at data privacy protection). Figure 2 gives a general overview of the INTEGRATE security platform.

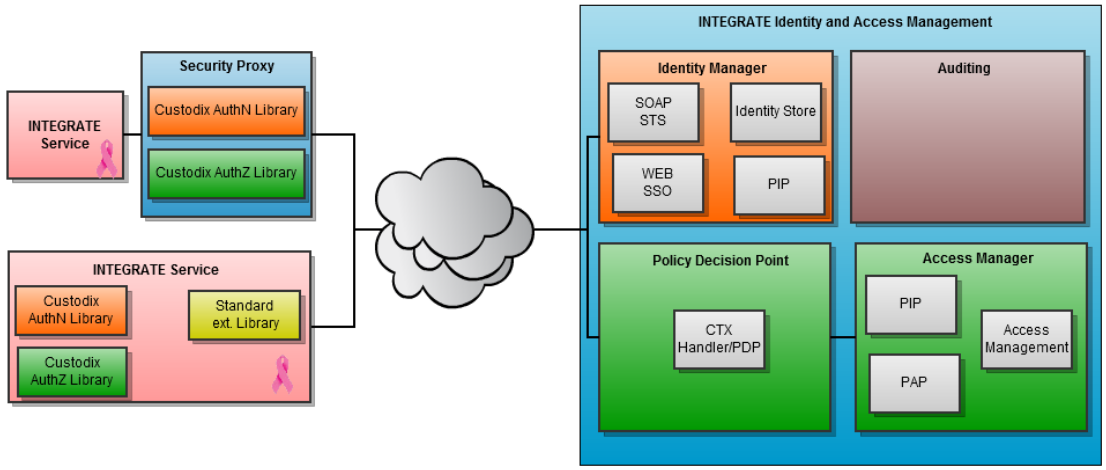


Figure 2: INTEGRATE Security Services

On the right side of the figure, the main platform services are listed:

- Identity Manager
  - The Identity Manager provides a front-end for user and service identity management and incorporates the modules for identity provision on a technical level, i.e. Identity Providers (IdPs) supporting Web Single Sign-On (SSO) and WS Security Token Service (STS) capabilities.
- Access Manager
  - The access management component is a management front-end that generates Access Control (AC) policies that are used by the authorisation services for evaluating access requests.
- Authorization Service
  - This service evaluates access request from all over the INTEGRATE infrastructure based on the security policies. The architecture and standards used allow this service to be easily implemented as a

distributed service (scalability). In the test-bed and pilot settings of INTEGRATE this is not expected to be necessary performance wise.

- Audit Service

Interaction between these services (security backend communication) are out of scope for integration with the framework which deals with interactions between the security infrastructure and service providers (SPs). A more elaborated description about these services can be found in Deliverables 2.4 and 2.6.

On the left side one finds the components that are available to Service Providers for integrating with the INTEGRATE security infrastructure.

2.2.2 Integration: Relevant Scenarios

Integrating a client or service provider with the INTEGRATE security infrastructure requires interfacing with the base security services at different "points" in the application-flow. This is best illustrated by describing three common communication scenarios depicted on Figure 3.

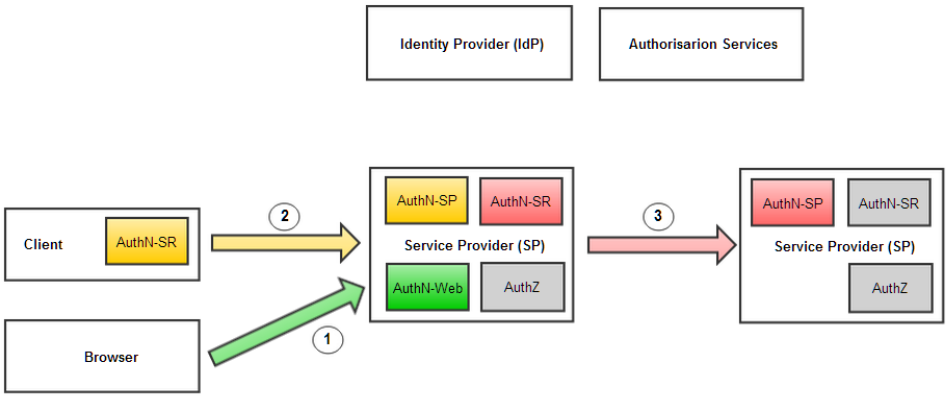


Figure 3: Communication scenarios relevant for security integration

2.2.2.1 Scenario 1: Browser communication

This scenario represents accessing a website which is part of the INTEGRATE environment. One concrete example of such a site in the project is the INTEGRATE Portal (which is a portlet technology based enterprise portal - Liferay[1]).

Web authentication in the INTEGRATE environment is based on the SAML 2.0 protocol and has been explained in detail in Deliverable 2.1 - State-of-the-Art Report on Standards. For developers of a web site who wish to integrate with INTEGRATE, this practically means that they need to be capable of consuming an INTEGRATE SAML 2.0 security token (containing security assertions).

The SAML Web Browser SSO[2] profile defines the SSO authentication protocol for a web site:

[1] http://www.liferay.com/

- SSO is initiated by a HTTP user agent which attempts to access a secured resource from a web site. If there is no active authenticated session yet on the web site for the given user agent, the web site should determine what identity provider (IdP) to use. How this is determined is out of scope of the profile. In Integrate we assume only one central IdP.
- The web site then issues a SAML authentication request. This request should be delivered through the user agent to the IdP. Several bindings such as HTTP Redirect and HTTP POST[3] define the communication between a web site and IdP for such an authentication request.
- If there is no active SSO session for the given user agent on the IdP yet, the IdP will initiate authentication. The SAML specification does not define how the IdP should authenticate the end user. E.g. the IdP can initiate authentication by presenting a username/password authentication form.
- Once there is an active SSO session the IdP will return, , an authentication assertion which identifies the end user through the user agent. For this IdP to web site communication of the assertion, the profile also defines multiple bindings such as HTTP POST or HTTP Artifact.
- The web site should then verify the assertion and create an authenticated session if the assertion is valid.

The Single Logout Profile defines how a principal can terminate the SSO session on the IdP and all active authenticated sessions on the different web sites the principal visited using SSO. Single logout can either be initiated from the IdP or from a web site.

The SAML bindings are defined in "Bindings for the OASIS Security Assertion Markup Language"[4]. E.g. the HTTP redirect binding defines how the SAML protocol messages (authentication requests, identity assertions, single logout requests ...) are transmitted through URL parameters. The HTTP POST binding on the other hand defines how those messages can be transmitted within the base-64 encoded content of an HTML form.

For minimal compliance a web site must support at least:

- the HTTP Redirect and HTTP POST bindings for web site to IdP communication of the Web Browser SSO Profile
- the HTTP POST and HTTP Artefact binding for IdP to web site communication.

A web site must also support the HTTP redirect and SOAP binding[5] of the Single Logout Profile, both for IdP and web site initiated single logout. A web site also needs to publish its SAML metadata. The metadata defines the SAML endpoints and the profiles and bindings supported by the site. Once a service is compliant it should register itself with the IdP by providing the IdP its metadata. Within INTEGRATE no service registration flow has yet been defined, therefore a service should register itself by mailing its metadata to integrate-support@custodix.com.

[2] The Web Browser SSO Profile is defined on page 14 of the specification "Profiles for the OASIS Security Assertion Markup Language (SAML)" (http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)
[3] http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
[4] http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
[5] http://www.w3.org/TR/soap/

```
<!-- Sample metadata of a compatible service provider -->
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="..."
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        ...
    </ds:Signature>
    <md:SPSSODescriptor AuthnRequestsSigned="true"
        WantAssertionsSigned="true"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:KeyDescriptor use="signing">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>...</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>...</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"
            Location="..." />
        <md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="..." />
        <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</md:NameIDFormat>
        <md:AssertionConsumerService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="..."
            index="0" isDefault="true" />
        <md:AssertionConsumerService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Artifact"
            Location="http..."
            index="1" isDefault="false" />
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

The AutnN-Web module, as shown on the picture, implements this minimal required functionality.

At any point of the application flow where an access control decision is required (which is not locally governed, but depends on global platform rules), a request for such a decision needs to be formulated towards the INTEGRATE authorisation services. On the picture, the “AuthZ” module represents the integration point which is responsible for dealing with creating such requests and interpreting AC decisions.

2.2.2.2 Scenario 2: Client to Service Provider (SP) communication

The second scenario illustrates the required integration when one wants to establish INTEGRATE compliant communication between a stand-alone client and an INTEGRATE SP. An example of this is the patient screening application as demonstrated during the first year project review.

In order to establish authenticated communication with a SP, service requestors (SRs) first need to obtain a security token from the Integrate Secure Token Service (STS) as defined in WS-Trust and WS-Security. With respect to integration, this means that client applications must support the WS-* stack (see "AuthN-SR" on Figure 3).
This implies that clients need to be able:

- to interpret the security requirements published by the SP (targeted STS, token encryption, token signing,..)
- to authenticate against the STS
- to send to security token to the SP

to handle authentication exceptions
At the receiving end, SP integration with the overall security architecture means that the SP should correctly advertise its security requirements and be able to accept and consume communicated security token ("AuthN-SP").

Authorisation is as described above.

2.2.2.3 Scenario 3: SP to SP communication

The final scenario deals with Service Provider (SP) to SP communication. Again integrating the application with the INTEGRATE authorisation infrastructure is the same as listed in scenario 1. The only difference compared to the previous scenarios, are the requirements demanded from the "AuthN-SR" block (at the service requestor). It not only needs to be capable of requesting security tokens from the STS for authenticating itself to the receiving SP, it should also support delegation (i.e. request a credential to act on behalf of another authenticated party). The latter means that the service requestor needs also integrate the necessary functionality to obtain delegation credentials from the STS.

2.2.3 Integration Approaches

The integration of security in an INTEGRATE service or client should be a straightforward process for the developer of this service or client. The integration effort should be as low as possible.

Integration with the security services can be done in different ways (i.e. at different logic layers). Integration can be done:

- Using INTEGRATE delivered modules and services
  - By integrating software modules (libraries) into the developed SP code.
  - By relying on a security gateway proxy service.
- By directly interfacing to the infrastructure security services

2.2.3.1 Integration with software modules (libraries)

A first solution offers to the developer a standard set of INTEGRATE security libraries that should be integrated in the client or web service code itself. These security

libraries provide the functionality to enable authentication and authorisation enforcement in the web services or clients.

Put in a simple way, this means that the modules described in 2.2.2 are delivered as packaged software components which can be easily incorporated into the developed application (e.g. through a generic API). The implementation of these modules is logically technology dependant (language, application environment, etc.). The different modules that currently are available to developers are listed in 2.2.4.1.

It is clear that it is infeasible in practice to provide packaged software for every technology platform.

2.2.3.2 Security gateway proxy service[6]

The least intrusive way and most convenient way of integration is by bundling the security functionality in a (reverse) proxy service. This means that SPs can be developed (more or less) without taking into account the INTEGRATE security environment. Security functionality is then provided by an independent application which proxies connections to the unsecured services. Towards service requestors, the new SP acts as a fully INTEGRATE compliant SP.

Working with a security gateway (proxy) is only possible to the extent that the covered security functionality can be completely separated from the SP application. This is in general not a problem for authentication, but only partially possible for authorisation. When the access control granularity cannot be limited to the level of the service call, tight coupling between the functional logic and authorisation logic is required.

Gateway Proxy for web sites

When used in front of a web site, the gateway will handle all SAML protocols for the web site. This implies that the gateway will publish SAML metadata, and that the gateway will redirect the user to the IdP when there is no active authenticated session.

[6] Note that research & development of a proxy service is outside of the scope of INTEGRATE, it is however developed in parallel in other projects, while leveraging on some of the developed INTEGRATE modules (esp. those for authorisation). If however proxy services are useful to the INTEGRATE tested and pilot environments and they are timely available, they will be used.

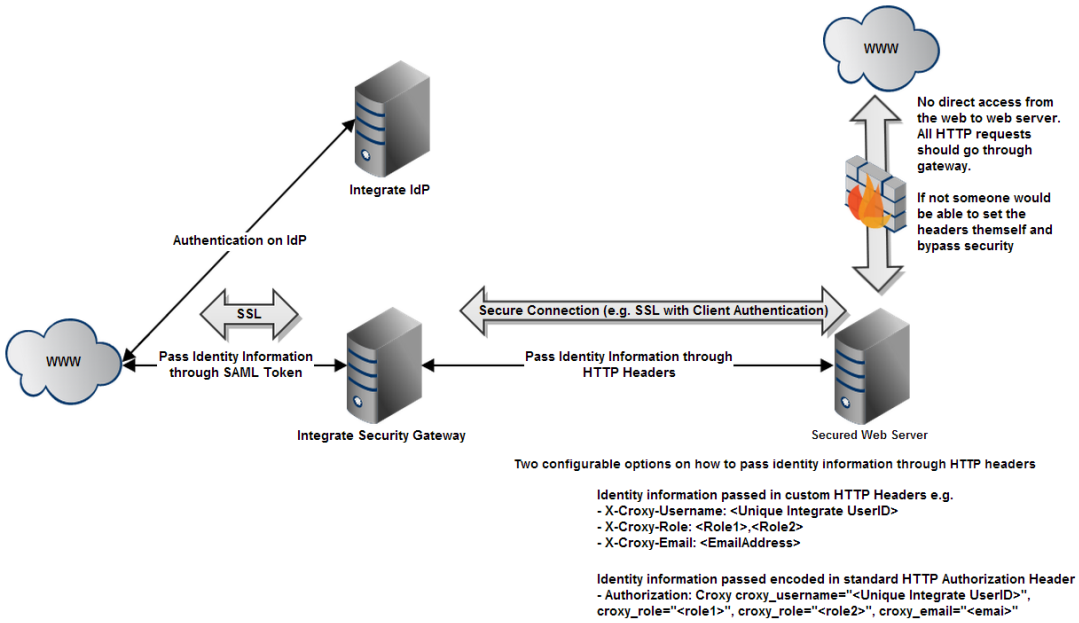


Figure 4 Security gateway proxy in front of a web service

The gateway will then interpret and verify the SAML identity token coming back from the IdP and set up an authenticated session. If there is an authenticated session, the gateway will insert an HTTP Authorization Header in each HTTP request forwarded to the web server. This HTTP Authorization header contains a comma separated list of name values pairs (Figure 4) extracted from the SAML token. The web site can then extract those attributes from the header and use them to identity the end user without having to worry about the SAML protocols.

2.2.3.3 Direct Interfacing to the Security Infrastructure

A final option for integrating clients and services in the INTEGRATE environment is by interfacing with the base components at the service level. In this case, application developers need to provide all security related application code themselves (e.g. a SAML consumer) and make sure that service requests to security service are performed compliant to the INTEGRATE architectural specification.

The architectural principles that INTEGRATE adheres to facilitate direct interfacing: loose coupling (separation of concerns) and the usage of commonly used standards (both on the communication as security level) such as SOAP-RPC, SAML, WS-Security, WS-Trust and XACML form the foundation, play a major role in making direct integration a viable solution in practice.

2.2.4 Security Integration Details

2.2.4.1 Available Software Modules

As mentioned in the previous section, the most convenient way of integrating is by relying on packaged software modules which encapsulate the interfacing with the INTEGRATE security components for application developers. The available (or planned) modules are listed below.

- AuthN-Web
  - Spring Security SAML extension.
    - A web site protected by Spring Security can implement the needed SAML profiles and binding by using and configuring the Spring Security SAML extension.
  - A liferay extension is available which integrate the Spring Security SAML extension into Liferay enabling SAML authentication.
  - SimpleSAML PHP[7]
- AuthN-SR
  - CXF: SOAP web service clients that are based on the Apache library CXF will automatically fetch a token from the Secure Token Service for web services configured as defined in WS-Trust and WS-Policy
- AuthN-SP
  - CXF implements SOAP web services in Java and supports the WS-* stack. Such a web service can then be configured to publish in their WSDL the security requirements (e.g. the STS from which clients should get a security token).
- AuthZ
  - Custodix Authz PEP Modules

2.2.4.2 Direct Interfacing to the Security Infrastructure

Because of the high adherence to standards during the design (and implementation) of the security architecture, direct interfacing remains a practically viable integration approach. On some technological platform a significant amount of the necessary functionality is available through standard libraries or through open source initiatives. In those cases, integration is often restricted to providing glue code (in those cases it is even debatable if encapsulating into a packaged module is even cost-efficient). A representative (non-exhaustive) list of such components is given at the end of this section. Note that some of these libraries are also used in the implementation of the INTEGRATE security services.

The following paragraphs contain the integration guidelines for direct interfacing. The components refer to the integration points illustrated on Figure 2.

AuthN-Web component (web browser communication)

The communication with INTEGRATE services from a web browser will use the single sign-on (SSO) profile of SAML. This profile does not have special integration requirements for the web browser itself.

AuthN-SP (WS, service provider) component

Each secured INTEGRATE web service (SOAP) will publish authN details through its WSDL by means of a WS-Policy (see Figure 5)

AuthN-SR (WS, service requestor) component

[7] http://simplesamlphp.org/

Each INTEGRATE service requestor (be it a web client or an SP that wants to initiate communication with another SP) will first need to find out which security requirement the relaying party imposes on communication (using the provided wsdl of the SP).

First of all the service requestor will need to obtain authentication credentials from the authentication service. Requests to the authentication service need to be structured according to security token service (STS) framework of WS-Trust (cf. RequestSecurityToken). Replies will be given according to the same standard (cf. RequestSecurityTokenResponse). This standard defines an XML format for exchanging STS requests and responses.

Calls to the authentication service are to be made as SOAP-RPC requests over SSL/TLS. The latter is used as a mechanism to ensure transport level confidentiality only. The (mutual) authentication capability of SSL/TLS is not used, trust is built relaying on the standard INTEGRATE mechanism SOAP-RPC: WS-Security[8].

The authentication tokens that are included in the responses of the STS are structured according to the SAML 2.0 Assertions definition.

Service Providers should also support delegation of security tokens. Interface wise this is not different from the earlier described process.

[8] Note that services need to be registered in the circle of trust (CoT) of the platform security services (IdP, authorisation service) in order to be able to access these.

```
<wsp:Policy wsu:Id="PrincipalServiceServiceSoapBindingPolicy" xmlns:sp="http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702" xmlns:t="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsaw="http://www.w3.org/2005/08/addressing" xmlns:wsp="http://www.w3.org/ns/ws-
policy" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex">
  <wsp:ExactlyOne>
    <wsp:All>
      <wsam:Addressing wsp:Optional="false">
        <wsp:Policy></wsp:Policy>
      </wsam:Addressing>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken>
                <wsp:Policy></wsp:Policy>
              </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128></sp:Basic128>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax></sp:Lax>
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp></sp:IncludeTimestamp>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:IssuedToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
            <sp:RequestSecurityTokenTemplate>
              <t:TokenType>urn:oasis:names:tc:SAML:2.0:assertion</t:TokenType>
              <t:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Bearer</t:KeyType>
            </sp:RequestSecurityTokenTemplate>
            <wsp:Policy>
              <sp:RequireInternalReference></sp:RequireInternalReference>
            </wsp:Policy>
            <sp:Issuer>
              <wsaw:Address>https://idp-dev-
integrate.custodix.com/sts/services/STS</wsaw:Address>
              <wsaw:Metadata>
                <wsx:Metadata>
                  <wsx:MetadataSection>
                    <wsx:MetadataReference>
                      <wsaw:Address>https://idp-dev-
integrate.custodix.com/sts/services/STS?wsdl</wsaw:Address>
                    </wsx:MetadataReference>
                  </wsx:MetadataSection>
                </wsx:Metadata>
              </wsaw:Metadata>
```

Figure 5: Ws-Policy authentication example

AuthN-SP (WS, receiver end) component

After the service requestor authenticated against the STS, the STS token is passed to the SP. The incoming token must be validated. Afterwards a security context can be set up in the SP by means of the issued security token. Next to this, it is possible that one or more attributes are included in the response from the STS, the SP must be able to deal with these attributes.

AuthZ component

Application developers are responsible of making calls to the authorisation service whenever an access control decision should be based on centrally governed policies.

Requests to the authorisation need to be structured according to OASIS XACML v2.0 standard using the INTEGRATE security vocabulary. Communication with the authorisation service should comply with the OASIS XACML v2.0 standard SAML 2.0 profile for authorisation decisions (cf. XACMLAuthzDecisionQuery). Replies will be given according to the same standard (cf. XACMLAuthzDecisionStatement).
Calls to the authorisation service are to be made as SOAP-RPC requests over SSL v3.0/TLS v1.3. The latter is used as a mechanism to ensure transport level confidentiality only. Instead of using mutual authentication, trust is built in by relying on message level security as defined in WS-Trust.
Service Providers requesting access control decisions should authenticate themselves to the Authorisation Service in the same way they would to other SPs, i.e. they need to obtain an authentication from the Identity Provider STS first and communicate that by embedding it into the service request (WS-Security). This mechanism was described in the previous sections.

Appendix: Relevant implementations

As mentioned, relying on commonly accepted standards means that one can rely on existing implementations. The paragraph below lists the most important libraries for two of the largest technological platforms.

Microsoft ".NET" environment

- Windows Communication Foundation[9] (WCF) is a bundled set of APIs included in the .NET framework for designing, implementing and deploying distributed applications. A WCF service exposes a contract via one or more endpoints. An endpoint has a URL (specifying where the endpoint can be accessed) and binding properties that specify how the data will be transferred. Next to services as REST and RSS, WCF offers an implementation of many of the WS-* standards. Using this implementation, SOAP-RPC requests can be generated over SSL/TLS. For enabling trust in the request a WS-Security implementation is available. All the WS-* implementations support interoperability with different platforms from WCF.
- Windows Identity Foundation[10] (WIF) is a framework for building identity-aware applications that abstracts the WS-Trust and WS-Federation protocols and presents developers APIs for building security token services (STS) and claims-aware applications. Trust between claim-aware application and STS can

[9] Window Communication Foundation: http://msdn.microsoft.com/en-us/library/dd456779.aspx
[10] Windows Identity Foundation: http://msdn.microsoft.com/en-us/security/aa570351.aspx

be established and identity delegation across the service boundaries is provided. For generating authentication tokens it offers an implementation of the SAML 1.1 and 2.0 standards.

Java environment

- Metro[11] is the Oracle web service stack. It offers APIs like JAX-WS and JAX-RS for developing web services. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP. Next to this, Metro implements many of the WS-* standards, like WS-Security and WS-Trust.
- CXF[12] is an opensource web service framework for Java based on the JAX-WS and JAX-RS interfaces. In functionality it is very similar to Metro. In contradiction to Metro CXF easily integrates with the Spring Framework
- OpenSAML[13] is an open source implementation of the Security Assertion Markup Language (SAML). It supports versions 1.0, 1.1 and 2.0. It deals with the issuing and validation of SAML authentication tokens.
- Sun XACML[14] is a java based open source implementation of the OASIS XACML standard, supporting version 2.0. It was once the de-facto reference implementation of the XACML standard. The implementation has not been maintained for a long time but is easily extendable and forms the basis for many other implementations, such as for example:.
    - JBoss XACML[15] is based on the Sun's XACML implementation.

2.3 Semantic integration guidelines

In the context of clinical trial research, data format and terminologies are usually established by the different institutions. Such heterogeneity increases the difficulty to understand that data by other members of the community. The INTEGRATE platform aims to achieve data integration by including a semantic interoperability layer [2]. This layer provides a homogenous interface in the retrieval of clinical data that is used by the different components of the platform. The functions of this layer can be mainly divided in two:

1. Extraction, Transformation and Load (ETL) of the information from the different data sources, i.e. to introduce new data sources into the Common Data Model (CDM). In the INTEGRATE project the CDM follows the HL7 Reference Information Model (RIM) [3].
2. Provide a common method to access the information stored in the CDM. Concepts requested within the queries require the same transformations performed during the integration of new data sources.

In both processes, concepts are transformed if needed in order to have a unique representation. In INTEGRATE the main terminologies used are SNOMED CT [4] and LOINC [5]. LOINC concepts do not require any transformation but SNOMED CT concepts require a transformation into the SNOMED normal form [6] (i.e. transforming

[11] Metro web service stack: http://metro.java.net/
[12] Apache CXF: http://cxf.apache.org/
[13] OpenSAML: https://wiki.shibboleth.net/confluence/display/OpenSAML/Home
[14] Sun XACML: http://sunxacml.sourceforge.net/
[15] Jboss XACML: https://community.jboss.org/wiki/PicketBoxXACMLJBossXACML

a concept into a normalized expression). These processes also assign a HL7 RIM class for those concepts (i.e. Terminology binding).
Once the information is stored, retrieval of information is carried out through the Semantic Interoperability Layer. In order to facilitate the composition of CIM-based queries, and to store additional data introduced by the users, a query builder module is available.

2.3.1 Semantic approach

As explained in the previous section the approach to achieve semantic integration is to include a layer in charge of solving the different problems due to the heterogeneity of terminologies and formats of information. The main components of this layer are shown in Figure 6.

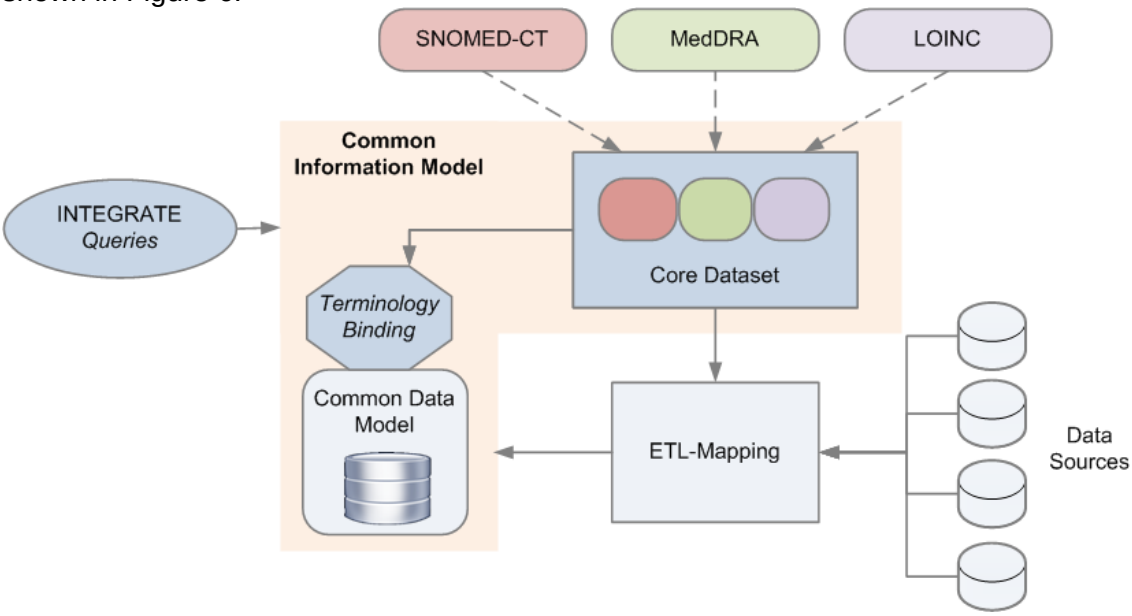


Figure 6: Semantic Layer structure

As presented in Figure 6 the main components of the layer are the ETL and the Common Information Model (CIM) which contains the core dataset, the terminology binding and the CDM. The core dataset interacts with the terminologies and with the ETL process to provide access to the terminology information in order to store the clinical data properly following the CDM. This component also receives the queries containing terms from a limited set of terminologies which are then translated to follow the model from the CDM.

The concepts are mapped to the core dataset, and then they need to be stored into a HL7 RIM class. The component that establishes such link is the Terminology Binding that follows the guidelines set in the HL7 Terminfo [7] project, to bind classes and attributes from the RIM to given concepts.

Finally, the CDM contains the data warehouse following the HL7 RIM. The queries are executed in the data warehouse so they need to follow the RIM structure as well. The components of the semantic layer are therefore used in two processes, the integration of new data sources and the access to homogenized data.

2.3.2 Integrating a new data source

The integration of a new data source into the system aims to provide a homogeneous interface for a large amount of complex information. This information can be encoded with different formats and may use different terminologies. In order to recover such data semantically, a transformation process is needed. However a first restriction is needed to ensure that the system is capable of interpreting the new information. In this case, the new data sources will have to be provided following the HL7 messaging standards. An example of the main components of these messages is provided in Figure 7.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:hl7-org:v3 CDA.xsd">
 ...
 <recordTarget>
   <patientRole>
     <id extension="db955e1e89d47f033496f5b0c3f367d3"
root="2.25.299518904337880959076241620201932965147.2.1" />
       <patient>
         <name/>
         <administrativeGenderCode code="F"
codeSystem="2.16.840.1.113883" />
         <birthTime value="19560310" />
       </patient>
   </patientRole>
 </recordTarget>
 ...
 <component>
   <structuredBody>
     <component>
       <section>
         ...
         <entry>
           <substanceAdministration classCode="SBADM" moodCode="EVN">
             <text>Doxorubicin</text>
              <consumable>
               <manufacturedProduct>
                <manufacturedLabeledDrug>
                 <code code="372817009"
codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
displayName="Doxorubicin (substance)" />
                </manufacturedLabeledDrug>
               </manufacturedProduct>
              </consumable>
             </substanceAdministration>
          </entry>
          ...
       </section>
     </component>
   </structuredBody>
 </component>
</ClinicalDocument>
```

Figure 7: HL7 Message

Such messages should include: (i) patients as entities and (ii) the corresponding patient interactions as Observations, Substance administration and Procedures among others.

Once the new data is received following the HL7 RIM, a study of that data is required as a difference in format may require a transformation in the ETL. The Extract and Transformation phases of the ETL are in charge of such modifications:

- Extraction of the data consists in reading the information from the data source and interpreting it into a recognizable structure which will be used through the rest of the process.
- Transformation of the data converts the previous data structure into a predefined model through a series of transformations following HL7 RIM guidelines. This process also includes small transformations to adapt the information to be usable by the final process of the ETL (e.g. unify date formats, scale transformations for units, etc.).
- Loading of the data loads the converted information during the process into a database allowing performing queries over that database. This database will follow the structure of the RIM model to allow the correct storage of the information previously transformed.

Besides the ETL there are more transformations required for the original data. Those transformations are part of the core dataset and are the transformation to the Normal Form for the concepts who are part of the SNOMED terminology and the terminology binding for all the concepts. The transformation to the SNOMED Normal Form allows having a common interpretation for the SNOMED concepts by transforming the concepts into a series of concepts and attributes increasing the granularity without losing information. The terminology binding helps to establish the correct class and attribute from the RIM to the medical concepts.

The CDM follows the HL7 RIM which provides an abstract model for representing health related information. The objective of the RIM is to build interoperability messages to communicate information systems, and it comprises classes with sets of attributes associated with one another by relationships. In the INTEGRATE project the model used is a subset of the entire RIM which allows to store the used information.

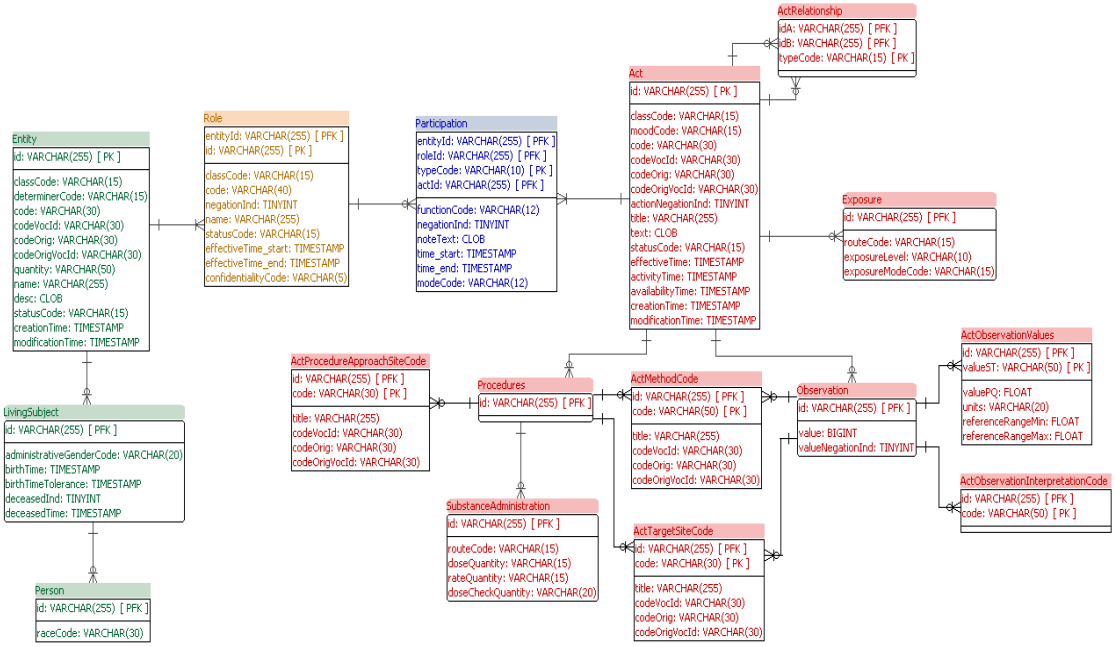


Figure 8: HL7 RIM-based schema

As described in Figure 8, the subset includes as main classes Act, Observation, Procedure, Substance Administration and Entity which are the classes of the RIM the concepts of the terminology can be translated to.

2.3.3 Accessing data (the semantic layer)

In order to access to the information from the CIM, this information must be first homogenized, according to the CDM and thus maintaining the consistency between the information that is being stored or retrieved. Data stored in CDM is accessed by an SPARQL [8] endpoint that retrieves the desired information from the CDM.
Semantic layer give the possibility to obtain a SPARQL template for a given core dataset concept. This template contains all the information necessary to retrieve data about the given concept. Building the template query for a given vocabulary concept, requires the interaction with the Core Dataset Service. The Core Dataset Service translates the concept into its normalized form that is the key for the extraction of the information implicit from the given concept, since each piece of information is stored by the CDM.

2.3.3.1 SPARQL Endpoint

The CDM developed in the INTEGRATE platform offers an SPARQL endpoint to query the data warehouse. This SPARQL endpoint is provided by a D2R Server [1]. The query is received by the semantic layer through an extended SPARQL language. These extensions introduce the possibility of expanding vocabulary concepts that appear in the original query. In this case, the query is expanded in the core dataset with the query expansion method. If it is not required to use the query expansion method [2], the original query is sent to the SPARQL endpoint of the D2R Server.

The query expansion method exploit information about relationships contained within the core dataset. If the query given to the semantic layer contains the function defined

in the query expansion method "*isAnysubClassOf*", the query is enriched with the vocabulary concepts that are related with the concept of the original query.

```
SELECT DISTINCT ?act_id WHERE {
        ?instParti a hl7rim:participation;
                hl7rim:participation_entityId ?id;
                hl7rim:participation_act ?instAct.
        ?instAct   hl7rim:act_classCode "SBADM";
                hl7rim:act_code ?code;
                hl7rim:act_id ?act_id.
FILTER (?code = isAnySubclassOf(108787006)).
FILTER (?id = "a459bca6dc09c131a8a78bd4b2574653").}
```

Epirubicin

```
… FILTER (?code = "108786002" || ?code = "108786002" || ?code =
"116079002" || ?code = "326830005" || ?code = "349848009" || ?code =
"35300007" || ?code = "68444001" || ?code = "326788003" || ?code =
"326789006" || ?code = "326796008" || ?code = "326802005" || ?code =
"326803000" …
```

Figure 9: Expanded SPARQL query on Anthracycline example

One example of this method is shown in the Figure 9. The query received by the semantic layer needs to be expanded with the Anthracycline concept and the results is an expanded query with all the concepts codes that are related with the Anthracycline code. Finally, this query has to be executed in the SPARQL endpoint to retrieve the information of the CDM.

2.3.4 Automatic query generation

As a mechanism to ease the process to query for the stored data, a new component has been included, the query builder. This component receives a series of concepts over which the user wants to perform a query. This concepts are transformed, normalizing them (if the concepts are from the SNOMED CT terminology) and having a RIM class assigned according to the terminology binding.

Once this process is completed a series of RIM classes and attributes is returned. These are used to search for a suitable query in the template library. This library includes templates for all the possible combinations of RIM classes and attributes that can be generated from any original concepts. An example of the first version of these templates is shown in the following figure.

```
<template version="0.3" id ="001" description="Template for querying
entities">
  <classes>
     <class>Entity</class>
  </classes>
  <rimClasses>
     <rimRelationship>
       <rimClass>Entity</rimClass>
       <rimAttribute>code</rimAttribute>
     </rimRelationship>
  </rimClasses>
  <inputConcept>
     %concept%
  </inputConcept>
  <normalizedOutput>
     %normalform%
```

```
  </normalizedOutput>
  <sqparqlQuery>
    <![CDATA[
       SELECT DISTINCT ?entityId ?birthTime $$optionalAttributes$$
       WHERE
       {
              ?entity       a hl7rim:entity;
                            hl7rim:entity_id %Entity_code%;
                            hl7rim:entity_id ?entityId;
                            hl7rim:entity_livingSubject ?subject.
              ?subject      hl7rim:livingSubject_birthTime ?birthTime

       OPTIONAL{
              ?subject      hl7rim:livingSubject_administrativeGenderCode
?gender.
       }
       }
    ]]>
  </sqparqlQuery>
  <optionalAttributes>
      <optionalAttribute>?gender</optionalAttribute>
  </optionalAttributes>
</template>
```

Figure 10: Template query for entities

In the template the main structure of the query is provided together with a series of optional attributes for which the users can ask. It also includes the information of the original concept and the normalized expression to ease the traceability of the process.

With these templates, the need for the user to memorize the available information is avoided and a common way to retrieve the information is established. The user can simply use the query included in the template with the optional attributes needed and perform the query as explained in the SPARQL endpoint section.

3 Adding new models / analytical tools

The main objective of the INTEGRATE Analysis Platform is to provide users with a web-based access to a collaborative, multi-functional and easy-to-use environment for exploiting, analysing and assessing the quality of large multi-level data. To achieve this, in a user friendly manner, the architecture of the platform is mainly divided into two parts:

1. A web-based interface that supports the interaction between the user and the functionality of the platform.
2. The back-end of the platform composed of the programming aspects of the different environments and languages adopted for implementing the platform's facilities, and the connectivity process which allows the interaction between these components.

The aforementioned architecture provides a flexible way of extending the functionality of the platform by plugging in new tools and models for statistical and predictive analysis, hiding at the same time the complexities of their computational infrastructure. On the other hand, a new analysis component should be designed following a precisely defined structure to achieve the communication between the tool/model and the platform's components, as depicted in the following figure.

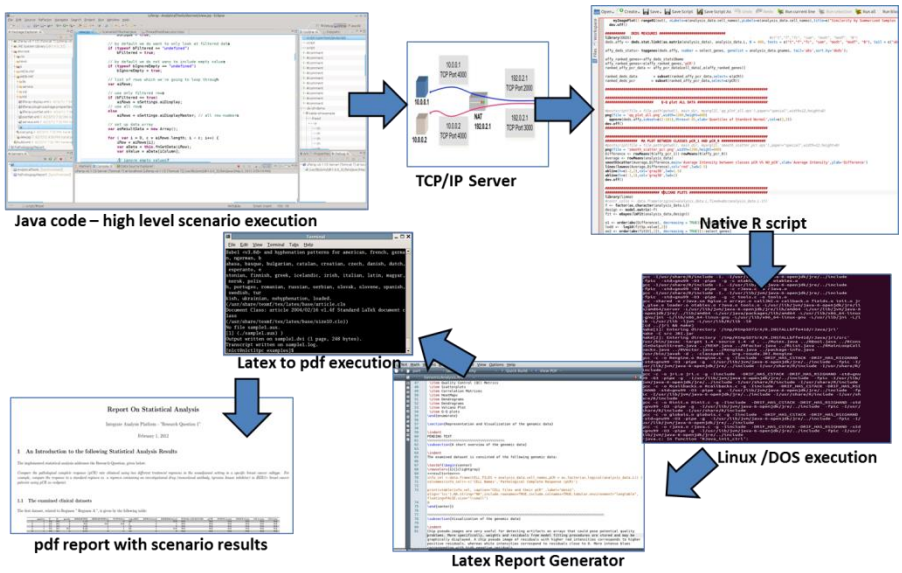


Figure 11 the back-end functionality of the Analysis Platform

Briefly presented, the front-end of the platform is based on the Liferay Portal[16] an enterprise web platform based on Java technologies that provides a consistent user interface for all the modules of the platform ensuring a seamless user experience. The front-end is coupled with an intuitive clinical data browser that takes as input query results from the INTEGRATE central datawarehouse, stores the data to the platform's database, and filters them according to any of the available parameters. Subsequently, the java interface is the intermediate component allowing the data being transferred from the database to the tools and models.

[16] Liferay (www.liferay.com)

The implementation of the statistical tools and predictive models is performed in R language[17].

```
svmFuncs <- caretFuncs #Default caret functions
svmFuncs$summary <-  function (data, lev = NULL, model = NULL)
{
  temp_var<-data.frame(pred=data[, "pred"],obs=data[, "obs"])

  acc_var<-((table(temp_var)[1,1]+table(temp_var)[2,2])/sum(table(temp_var)))

  require(pROC)
  if (!all(levels(data[, "pred"]) == levels(data[, "obs"])))
    stop("levels of observed and predicted data do not match")
  rocObject <- try(pROC:::roc(data$obs, data[, lev[1]]), silent = TRUE)
  rocAUC <- if (class(rocObject)[1] == "try-error")
    NA
  else rocObject$auc

  out <- c(rocAUC,acc_var,sensitivity(data[, "pred"], data[, "obs"],lev[1]), specificity(data[, "pred"], data[, "obs"], lev[2]),
           posPredValue(data[, "pred"], data[, "obs"]),negPredValue(data[, "pred"], data[, "obs"]),
           (posPredValue(data[, "pred"], data[, "obs"])*sensitivity(data[, "pred"], data[, "obs"],lev[1]))/(posPredValue(data[, "pred"]
  names(out) <- c("auROC","Accuracy","Sensitivity", "Specificity","Precision","NPV","F_measure")

  out
}

svmFuncs$pred
function (object, x)
{
  tmp <- predict(object, x)
  if (object$modelType == "Classification" & modelLookup(object$method)$probModel[1]) {
    out_p <- cbind(data.frame(pred = tmp), as.data.frame(predict(object,
```

Figure 12 A piece of software R code

To facilitate embedding R functionality in our java-based interface, a client/server concept using TCP/IP protocol[18] was used for the communication between the R system and the end-user allowing the interaction between the platform and the execution framework. The platform supports an engine[19] to create dynamically statistical analysis reports by enabling integration of R code and Latex documentation[20]. On-the-fly reporting is therefore generated by combining the programming source code and the corresponding documentation into a single file.

By following the steps described below, one can extend the functionality of the platform with new tools and models. All of the elements described below, should be provided as a unique framework to an administrator, in order to upload and make a new tool or model functional in the platform.

3.1.1 The software language of a new tool or model

The software language, adopted for the needs of the tools and models implementation is R. The reason for choosing R is that it is a free software environment that compiles and runs on a variety of platforms, has a wide community of researchers and developers who share open-source documentation and software, is a highly extensible coherent system for software development, and a good connectivity with other software environments. In many cases, the software code of a tool or a model is supported by publicly available libraries, found in “R packages”. R packages are installed into libraries, which are directories in the file system containing a subdirectory for each package installed there.

The INTEGRATE Analysis Platform uses a pool of R packages, previously installed to the system that hosts the platform. Their installation is performed via R code that runs at the first time that the platform’s software is set up. The R packages can be installed

[17] The R project for Statistical Computing (www.r-project.org)
[18] Rserve, a binary R server www.rforge.net/Rserve
[19] The Sweave tool (www.statistik.lmu.de/~leisch/Sweave)
[20] Latex, a document preparation system (www.latex-project.org)

either from a local place (stored as binary files) or directly from the web repositories. When a tool or model is executed, the libraries are called via its software code. In order to avoid direct package installation failures from the web repositories due to connectivity issues via the web, packages are first downloaded locally to the system and then installed by executing R code. Therefore, in case a new tool or model is to be plugged in the platform the following pipeline procedure has to be followed:

1. If the tool or model has dependencies to libraries from R packages, the user must provide their binary files.
2. The user provides the platform with R code for installing the packages to the system.
3. The user provides the platform with R code of the new tool or model.

3.1.2 Interaction between a new tool or model and Latex

An important part of the INTEGRATE platform is the way it provides the analysis results from a tool or model, where dynamically statistical and predictive analysis reports are created by enabling integration of R code and Latex documentation. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it. The Latex document file is then called from the R script that implements the tool or model, and on-the-fly reports in both pdf and html format are generated and stored as metadata information of the analysis. Thus, a new tool or model must provide its analysis results in a dynamically way through a Latex documentation file that interacts with its core R software.

3.1.3 Interaction between a new tool or model and the java front-end

The front-end of the INTEGRATE platform is the main connectivity component between the user, the software code and the data warehouse. Through the platform's interface the user selects the data for examination, the performed analysis, and gets access to the analysis results. The prerequisites to the way the described connectivity between a new tool and model will be established successfully are described below.

- Determine the variables that the tool or model takes as input arguments.
- Provide metadata information for the required input arguments (i.e. a column with clinical characteristics and a column with regimen names).
- Information for creating the necessary widget controls for interaction with the user and their proper assignment in the layout of the platform, grouped in a single container.
- Creation of the necessary handling servlet (or modification of an existing one) that will serve as bridge among the interface and the processing modules.

A representative example is given by the following figure. The user has implemented software code that performs survival analysis and gives a brief description of the new tool. An important step is to determine the required variables-parameters that feed the tool. In that case, the user can either select the appropriate variables from the existed database or provide their unique terms as defined by healthcare terminology organizations like SNOMED. Optionally, a text field listing useful notes content for running the tool is available.

The user uploads the necessary software scripts responsible for installing, if required, the libraries needed for the successful implementation of the analysis, running the analysis, generating the pdf analysis report and the html reporting, respectively. The last step is to upload the libraries' binary files, having dependencies with the user's software code.

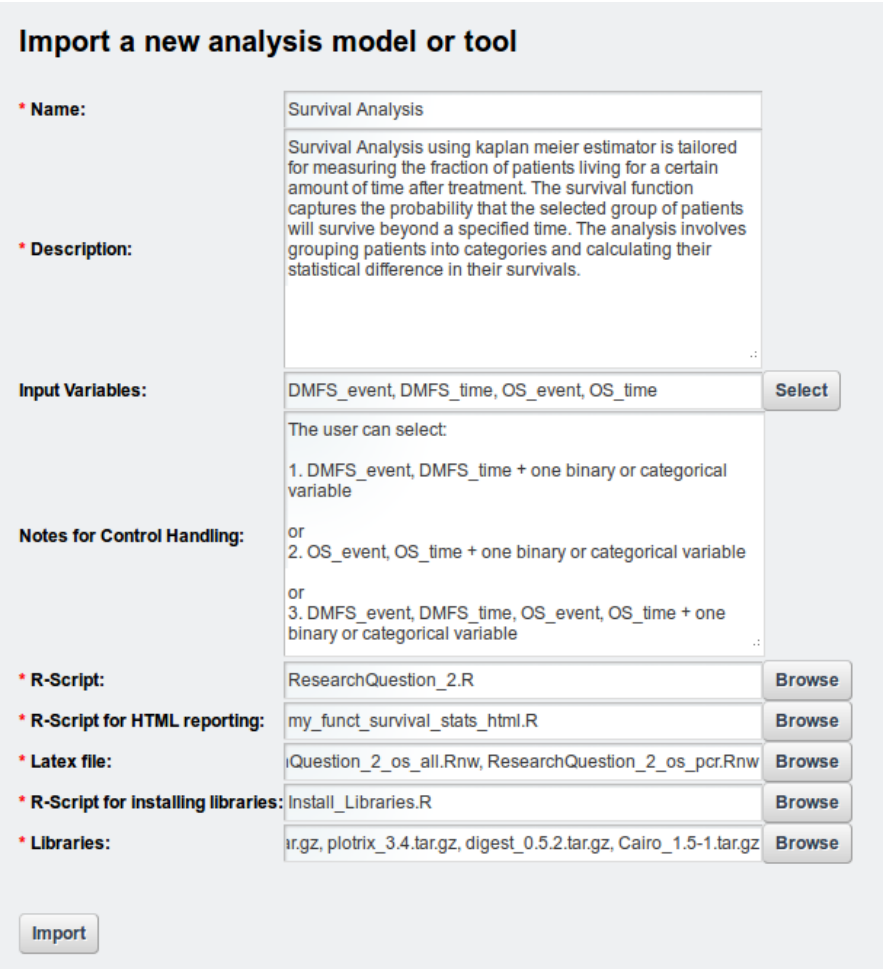


Figure 13 Platform's service layout for adding a new statistical tool or model

4 Conclusion

The integration guidelines covered in this document describe how to integrate new data sources, new components and new models/analytical tools in the INTEGRATE platform. The main goal of specifying these guidelines is to provide uniform interfaces that enable efficient accessing data and knowledge between the different INTEGRATE tools and services. These guidelines can be used as internal validator of the conformance of the tools and services developed in the architecture.

5 REFERENCES

[1] Bizer C and Cyganiak R. 2006. D2R Server – Publishing relational databases on the semantic web. In the 5th International Sematic Web Conference (ISWC).

[2] Paraiso S et al. Semantic interoperability solution for multicentric breast cancer trials at the INTEGRATE EU Project. In Proceedings of the 6th International conference in Health Informatics 2013.

[3] Beeler G, Case J, Curry J, Hueber A, Mckenzie L, Schadow G, Shakir AM. HL7 Reference Information Model. 2003.

[4] Donnelly K, SNOMED-CT: The Advanced Terminology and Coding System for eHealth. Medical and care compunetics 3 2006: 279-290.

[5] McDonald CJ, Huff SM, Suico JG, Hill G, Leavelle D, Aller R, Forrey A, Mercer K, DeMoor G, Hook J, Williams W, Case J, Maloney P. LOINC, a Universal Standard for Identifying Laboratory Observations: A 5-Year Update. Clinical Chemistry. 2003; 49 No. 4: 624-633.

[6] College of American Pathologists. SNOMED Clinical Terms Guide. Transforming Expressions to Normal Forms. August 2006. www.cap.org/apps/docs/snomed/documents/transformations_to_normal_forms.pdf

[7] Cheetham E, H. Dolin R, Markwell D, Curry J, Gabriel D, Hausam R, Knight B, Rector A, Spackman K, Townend I. Using SNOMED CT in HL7 v3 Implementation Guide, Release 1.5. 2008.

[8] Prud'hommeaux Eric, Seaborne Andy. SPARQL Query Language for RDF. January 2008. http://www.w3.org/TR/rdf-sparql-query/