

ReMINE

High performances prediction, detection and monitoring
platform for patient safety risk management

FP7 Contract: 216134



– Deliverable –

D.3.6 Third Revision of the WP3 framework

Document Information

Document Name: ReMINE_D3.6_IW_WP3

Revision: V0.4

Revision Date:

Author: Adrian Podea (Info World)

Contributors: INDRA

Security: Confidential (Consortium Only)

Document Information

Consortium, European Commission

Approvals

	Name	Beneficiary	Date	Visa
<i>Project Coordinator</i>	Michele CARENINI	NOEMALIFE		
<i>Technical Manager</i>	Simone RONDENA	NOEMALIFE		

Document History

Revision	Date	Modification	Author
Version 0.1	21/01/2011	ToC first draft	IW
Version 0.2	28/01/2011	Chapter 1,2,3	IW
Version 0.3	02/02/2011	Chapter 4,5	IW
Version 0.4	04/02/2011	Indra contribution	INDRA
Version 0.5	04/02/2011	Conclusions and final review	IW

Table of Contents

Table of Contents	3
1. Executive summary.....	4
2. Introduction.....	4
2.1. Deliverable vs. Project objectives	4
2.2. Deliverable structure and Partners contribution	4
3. Architecture overview for WP3 components.....	5
4. WP3 Services enhancements	6
4.1. Auditing service.....	6
4.1.1. Introduction	8
4.1.2. Business scenarios and use cases	9
4.1.3. Detailed functional model	10
4.1.4. Data contracts.....	11
4.1.5. Data definitions.....	13
4.1.6. Audit implementation in WP3 Services	30
4.2. RLUS WS-Eventing.....	33
4.2.1. Architecture overview	34
4.2.2. Service contracts.....	35
4.2.3. Detailed functional model on each interface	35
4.2.4. Data contracts.....	39
4.2.5. Database layer	40
5. Metadatabase reasoner	43
6. RLUS Wrapper for process mapper	62
7. Conclusion	65
8. Glossary	65

1. Executive summary

Deliverable D.3.6 Third Revision of the WP3 framework represents the last deliverable from task T3.2 – Metadatabase semantic data integration definition and development. This succeeds a list of previous deliverables related to task T3.2:

- D.3.5 First Revision of the WP3 framework
- I.D. First Revision of the WP3 framework

This deliverable aims at briefing the modifications and development that took place from the 2nd revision of the WP3 services in order to achieve the objectives listed in task T3.2: developing the Metadatabase structure. Modifications explained in this deliverable can be seen as enhancements to the developed WP3 services and the reasoner based on HL7 Templates and ADL.

2. Introduction

2.1. Deliverable vs. Project objectives

Task 3.2 Metadatabase semantic data integration definition and development – Objectives:

This task aims at developing a metadatabase structure where all the REMINE information will focus on following aspects:

- To develop Incident alerting real time and Post process Incident alerting based on mining technique
- To develop a knowledge extraction system based on REMINE metadatabase
- Integration with wp2 (Data Capture and RAPS alerting), wp4(RAPS Management process Support) and wp5 (Info Broker Patient Safety Framework)

2.2. Deliverable structure and Partners contribution

Chapters	Responsible partner
1. Executive summary	IW
2. Introduction	IW
3. Architecture overview	IW
4. WP3 Services enhancements	IW
5. Metadatabase reasoner	IW

6. RLUS Wrapper for process mapper	INDRA
7. Conclusions	ALL
8. Glossary	ALL

3. Architecture overview for WP3 components

Figure 1 represents the WP3 place in the data-flow architecture of the whole ReMINE project. As seen WP3 consists of three different components: the Process Mapper, the Database (database service with reasoner) and Data Mining & Knowledge Inference.

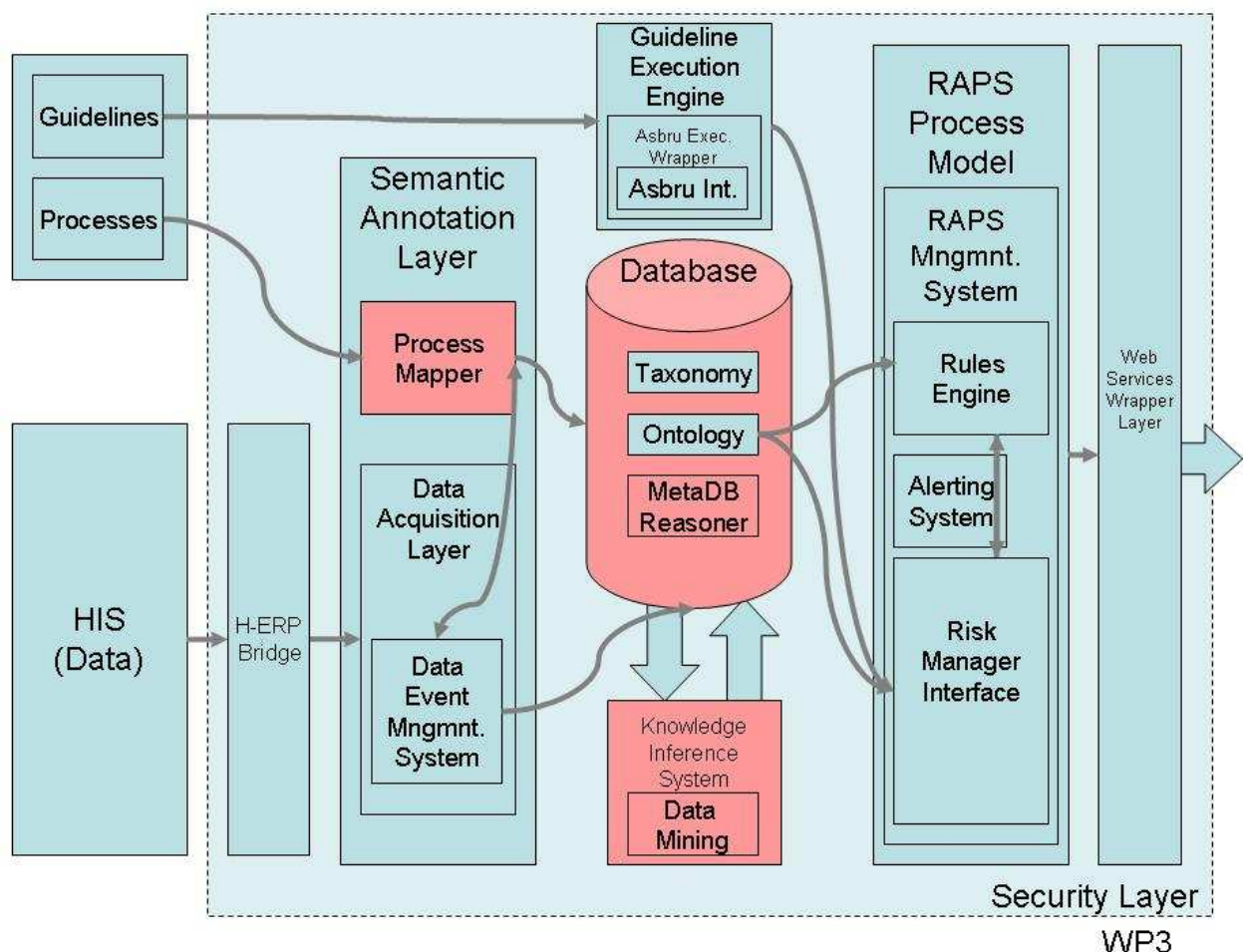


Figure 1 – The place of WP3 in ReMINE architecture

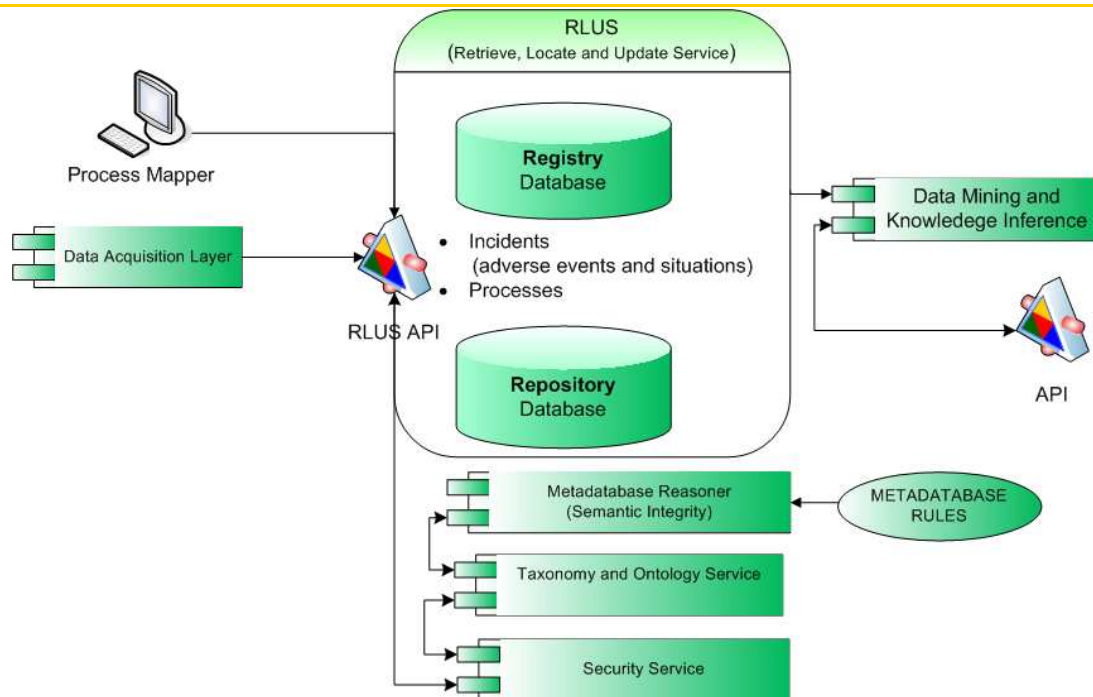


Figure 2 – Detailed WP3 Architecture

The Process Mapper (Intalio) in the semantic annotation layer will store processes in the database through the Metadatabase Service (RLUS) while exchanging messages with the Data Event Management System. RLUS provides the API for the Database acting like a file system allowing storing many types of resources with appropriate metadata. Metadatabase reasoner (based on Archetype Definition Language) will check the data for semantic integrity before stored in the Database. The Data-Mining process is responsible for applying specific algorithms for knowledge discovery from existing data and for transferring this knowledge to the Database in the form of 'rules', intelligent system parameters and trained models.

4. WP3 Services enhancements

4.1. Auditing service

Audit trail is one the functionalities of the Security Services that was finish for the 3rd revision of the WP3 services. The security services for the metadatabase will be based on well-established standards and practices such as:

- LDAP protocol;
- WS-Security specification;
- X509 certificates.

These services will be in charge of providing for the following 'functionalities':

- CIA:
 - o Confidentiality – encrypted message.

- Integrity – message hasn't been tampered.
 - Authentication – prove identity.
- Authorization – role based access.
- Accountability – audit trail.
- Policies – mutually agreed by involved parties.

From the client application (RLUS, TOS) perspective, the security services are in charge of:

- Authentication:
 - SAML assertions verified by the called service.
- Role based authorization:
 - Roles stored in LDAP.
 - Policies defined using XACML language.
- Record level authorization.
- Audit trail.

The audit services provide the means to address the issues of liability management, asset protection and quality of service. To facilitate a timely response to policy violations, security incidents or infrastructure and application failures, the metadatabase system supports monitoring, logging, analyzing, and reporting on every level of its architecture.

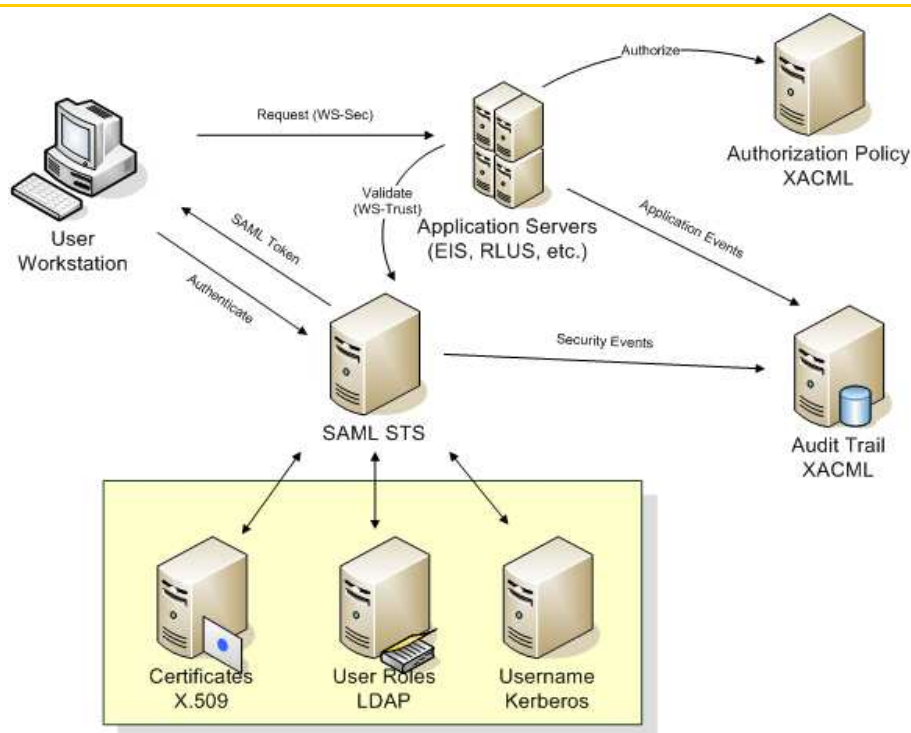


Figure 3 – Security services architecture

4.1.1. Introduction

For implementation modality the description will follow contracts between the service and its clients:

- Service Contract: the programming interfaces published by this service: what every method does, what in/out parameters has and what exceptions throws
- Data Contract: the classes this service use and the Class Diagram
- Fault Contract: the exceptions' list this service throws.

This service will facilitate the storage of audit messages sent by other services/applications into a database, replacing the actual message logging service which stores the collected data in a text file. The service will catch all usage data like who accessed healthcare data, when, for what action, from where, and which patients' records were involved, to help assure healthcare privacy and security in ReMINE platform.

In data definitions subchapter this document specifies audit data to be collected and communicated from automated systems. For each audited event, this document specifies the minimal data requirements plus optional data for the following event categories:

1. **Security administrative events** - establishing and maintaining security policy definitions, secured object definitions, role definitions, user definitions, and the relationships among them. In general, these events are specific to the administrative applications.

2. **Audit access events** - reflecting special protections implemented for the audit trail itself.
3. **Security-mediated events** - recording entity identification and authentication, data access, function access, non-repudiation, cryptographic operations, and data import/export for messages and reports. In general, these events are generic to all protected resources, without regard to the application data content.

Patient care data events - documenting what was done, by whom, using which resources, from what access points, and to whose medical data. In general, these audits are application-specific since they require knowledge of the application data content

4.1.2. Business scenarios and use cases

1) Catch and store the message in the database.

Section	Description
Business name	Store message.
Summary	The main goal of the service is to catch any message sent by the application, break it into atomic elements and store the message in a database for a better management of auditing. The Audit Service must treat every message in a uniform way regarding of its sender.
Triggers	The service must receive a message to store in the database.
Basic course of events	<ol style="list-style-type: none">1. One of the applications sends a message.2. The Audit service listens and catches the message.3. The message is processed and broken into atomic elements.4. The message is written in the database.
Alternative paths	In case the message can't be written into the database, the service keeps listening for other messages and doesn't report any error.
Postconditions [Optional]	The service keeps listening for other messages.

2) Quick search based on II (HL7 II data type)

Section	Description
Business name	Simple search
Summary	It is possible to query the database and find all the messages regarding a specific encounter by supplying its II (partially or fully). The result will be a set of records representing all the messages referring the specified encounter.
Triggers	The user wants to find all the messages regarding a specific encounter and must provide its extension and/or root.

Section	Description
Basic course of events	<ol style="list-style-type: none"> 1. The user enters the II (extension and/or root) 2. The database is queried. 3. The messages referring to the specified encounter are returned.
Alternative paths	<p>If the user doesn't enter an II, the query returns all the logged messages.</p> <p>If the user enters at least the root or the extension then the search is performed.</p> <p>If the search doesn't return any results then an adequate information may be shown to the user.</p>
Postconditions [Optional]	

4.1.3. Detailed functional model

Audit trail service will expose two contracts reach with one method:

- a) IMessageQuery with QuickSearch method
- b) MessageStore with StoreMessage method

StoreMessage method:

Business friendly name [Mandatory]	Store Message
Description [Mandatory]	Allows the service to catch and store the messages in the database.
Precondition [Mandatory]	Some other application/service must send a message.
Inputs [Mandatory]	Exception Message or a Business Message
Outputs [Mandatory]	None
Invariants [Optional]	N/A
Postconditions [Optional]	N/A
Exception Conditions [Mandatory]	No exceptions are thrown.
Aspects left to implementer [Mandatory]	N/A
Relationship to levels of conformance [Optional]	N/A
Notes [Optional]	N/A

Other relevant content [Optional]	N/A
--	-----

QuickSearch method: The IMessageQuery will be responsible for querying the database (simple and quick search by ID, an extended search by some different parameters, message get).

Business friendly name [Mandatory]	Quick Search
Description [Mandatory]	Allows the user to search by some simple criteria (e.g. II – extension and root)
Precondition [Mandatory]	The user enters an extension and/or root to search for.
Inputs [Mandatory]	The II value to search for
Outputs [Mandatory]	Found records.
Invariants [Optional]	N/A
Postconditions [Optional]	N/A
Exception Conditions [Mandatory]	No exceptions are thrown.
Aspects left to implementer [Mandatory]	N/A
Relationship to levels of conformance [Optional]	N/A

4.1.4. Data contracts



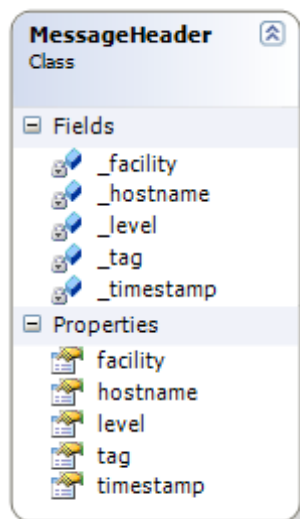


Figure 4 – Audit Service Data Contracts

4.1.5. Data definitions

The proposed data elements are grouped into these categories:

1. Event Identification - what was done
2. Active Participant Identification - by whom
3. Network Access Point Identification - initiated from where
4. Audit Source Identification - using which server
5. Participant Object Identification - to what record

1. Event Identification

The following data identifies the name, action type, time, and disposition of the audited event. There is only one set of event identification data per audited event.

a) Event ID

Description

Identifier for a specific audited event, e.g., a menu item, program, rule, policy, function code, application name, or URL. It identifies the performed function.

Optionality: Required

Format / Values

Coded value, either defined by the system implementers or as a reference to a standard vocabulary. The "code" attribute must be unambiguous and unique, at least within Audit Source ID (see section 5.4).

Examples of Event IDs are program name, method name, or function name. For implementation defined coded values or references to standards, the XML schema defines these optional attributes:

<i>Attribute</i>	<i>Value</i>
CodeSystem	OID reference
CodeSystemName	Name of the coding system; strongly recommended to be valued for locally-defined code-sets
DisplayName	The value to be used in displays and reports
OriginalText	Input value that was translated to the code

To support the requirement for unambiguous event identification, multiple values may not be specified.

Rationale

This identifies the audited function. For "Execute" Event Action Code audit records, this identifies the application function performed.

b) Event Action Code

Description

Indicator for type of action performed during the event that generated the audit.

Optionality: Optional

Format / Values

Enumeration:

<i>Value</i>	<i>Meaning</i>	<i>Examples</i>
C	Create	Create a new database object, such as Placing an Order
R	Read/View/Print/Query	Display or print data, such as a Doctor Census
U	Update	Update data, such as Revise Patient Information
D	Delete	Delete items, such as a doctor master file record
E	Execute	Perform a system or application function such as log-on, program execution, or use of an object's method

Rationale

This broadly indicates what kind of action was done on the Participant Object.

Notes

Actions that are not enumerated above are considered an Execute of a specific function or object interface method or treated two or more distinct events. An application action, such as an authorization, is a function Execute, and the Event ID would identify the function.

For some applications, such as radiological imaging, a Query action may only determine the presence of data but not access the data itself. Auditing need not make as fine a distinction.

Compound actions, such as "Move," would be audited by creating audit data for each operation - read, create, delete - or as an Execute of a function or method.

c) Event Date/Time

Description

Universal coordinated time (UTC), i.e., a date/time specification that is unambiguous as to local time zones.

Optionality: Required

Format / Values

A date/time representation that is unambiguous in conveying universal coordinated time (UTC), formatted according to the ISO 8601 standard [ISO8601]

Rationale

This ties an event to a specific date and time. Security audits typically require a consistent time base, e.g., UTC, to eliminate time-zone issues arising from geographical distribution.

Notes

In a distributed system, some sort of common time base, e.g., an NTP [RFC1305] server, is a good implementation tactic.

d) Event Outcome Indicator

Description

Indicates whether the event succeeded or failed.

Optionality: Required

Format / Values

Enumeration:

Value	Meaning
0	Success
4	Minor failure; action restarted, e.g., invalid password with first retry
8	Serious failure; action terminated, e.g., invalid password with excess retries
12	Major failure; action made unavailable, e.g., user account disabled due to excessive invalid log-on attempts

Rationale

Some audit events may be qualified by success or failure Indicator. For example, a Log-on might have this flag set to a non-zero value to indicate why a log-on attempt failed.

Notes

In some cases a "success" may be partial, for example, an incomplete or interrupted transfer of a radiological study. For the purpose of establishing accountability, these distinctions are not relevant.

e) Event Type Code

Description

Identifier for the category of event.

Optionality: Optional

Format / Values

Coded value enumeration, either defined by the system implementers or as a reference to a standard vocabulary. For implementation defined codes or references to standards, the XML schema defines these optional attributes:

Attribute	Value
CodeSystem	OID reference
CodeSystemName	Name of the coding system; strongly recommended to be valued for locally-defined code-sets
DisplayName	The value to be used in displays and reports
OriginalText	Input value that was translated to the code

Since events may be categorized in more than one way, there may be multiple values specified.

Rationale

This field enables queries of messages by implementation-defined event categories.

2. Active Participant Identification

The following data identify a user for the purpose of documenting accountability for the audited event. A user may be a person, or a hardware device or software process for events that are not initiated by a person. Optionally, the user's network access location may be specified. There may be more than one user per event, for example, in cases of actions initiated by one user for other users, or in events that involve more than one user, hardware device, or system process. However, only one user may be the initiator/requestor for the event.

a) User ID

Description

Unique identifier for the user actively participating in the event

Optionality: Required

Format / Values

User identifier text string from the authentication system. It is a unique value within the Audit Source ID

Rationale

This field ties an audit event to a specific user.

Notes

For cross-system audits, especially with long retention, this user identifier will permanently tie an audit event to a specific user via a perpetually unique key. For node-based authentication -- where only the system hardware or process, but not a human user, is identified -- User ID would be the node name.

b) Alternative User ID

Description

Alternative unique identifier for the user

Optionality: Optional

Format / Values

User identifier text string from authentication system. This identifier would be one known to a common authentication system (e.g., single sign-on), if available.

Rationale

In some situations a user may authenticate with one identity but, to access a specific application system, may use a synonymous identify. For example, some "single sign on" implementations will do this. The alternative identifier would then be the original identify used for authentication, and the User ID is the one known to and used by the application.

c) User Name

Description

The human-meaningful name for the user

Optionality: Optional

Format / Values

Text string

Rationale

The User ID and Alternative User ID may be internal or otherwise obscure values. This field assists the auditor in identifying the actual user.

d) User Is Requestor

Description

Indicator that the user is or is not the requestor, or initiator, for the event being audited.

Optionality: Optional

Format / Values

Boolean, default/assumed value is "true".

Rationale

This value is used to distinguish between requestor-users and recipient-users. For example, one person may initiate a report- output to be sent to another user.

e) Role ID Code

Description

Specification of the role(s) the user plays when performing the event, as assigned in role-based access control security.

Optionality: Optional; multi-valued

Format / Values

Coded value, with attribute "code" valued with the role code or text from authorization system. More than one value may be specified. The codes may be implementation-defined or reference a standard vocabulary enumeration. For implementation defined codes or references to standards, the XML schema defines these optional attributes:

<i>Attribute</i>	<i>Value description</i>
CodeSystem	OID reference
CodeSystemName	Name of the coding system; strongly recommended to be valued for locally-defined code-sets
Display Name	The value to be used in displays and reports
OriginalText	Input value that was translated to the code

Rationale

This value ties an audited event to a user's role(s). It is an optional value that might be used to group events for analysis by user functional role categories.

Notes

Many security systems are unable to produce this data, hence it is optional. For the common message, this identifier would be the one known to a common authorization system, if available. Otherwise, it is a unique value within the Audit Source ID (see section 5.4). Consider using a globally unique identifier associated with the role to avoid ambiguity in auditing data collected from multiple systems. Role ID is not a substitute for personal accountability. Ambiguities arise from composite roles and users with multiple roles, i.e., which role within a composite is being used or what privilege was a user employing?

3. Network Access Point Identification

The network access point identifies the logical network location for application activity. These data are paired 1:1 with the Active Participant Identification data.

a) Network Access Point Type Code

Description

An identifier for the type of network access point that originated the audit event.

Optionality: Optional

Format / Values

Enumeration:

<i>Value</i>	<i>Meaning</i>
1	Machine Name, including DNS name
2	IP Address
3	Telephone Number

Rationale

This datum identifies the type of network access point identifier of the user device for the audit event. It is an optional value that may be used to group events recorded on separate servers for analysis of access according to a network access point's type.

b) Network Access Point ID

Description

An identifier for the network access point of the user device for the audit event. This could be a device id, IP address, or some other identifier associated with a device.

Optionality: Optional

Format / Values

Text may be constrained to only valid values for the given Network Access Point Type, if specified. Recommendation is to be as specific as possible where multiple options are available.

Rationale

This datum identifies the user's network access point, which may be distinct from the server that performed the action. It is an optional value that may be used to group events recorded on separate servers for analysis of a specific network access point's data access across all servers.

Note

Network Access Point ID is not a substitute for personal accountability. Internet IP addresses, in particular, are highly volatile and may be assigned to more than one person in a short time period.

Examples

Network Access Point ID: SMH4WC02

Network Access Point Type: 1 = Machine Name

Network Access Point ID: 192.0.2.2

Network Access Point Type: 2 = IP address

Network Access Point ID: 610-555-1212

Network Access Point Type: 3 = Phone Number

4. Audit Source Identification

The following data are required primarily for application systems and processes. Since multi-tier, distributed, or composite applications make source identification ambiguous, this collection of fields may repeat for each application or process actively involved in the event. For example, multiple value-sets can identify participating web servers, application processes, and database server threads in an n-tier distributed application. Passive event participants, e.g., low-level network transports, need not be identified.

Depending on implementation strategies, it is possible that the components in a multi-tier, distributed, or composite applications may generate more than one audit message for a single application event. Various data in the audit message may be used to identify such cases, supporting subsequent data reduction. This document anticipates that the repository and reporting mechanisms will perform data reduction when required, but does not specify those mechanism.

a) Audit Enterprise Site ID

Description

Logical source location within the healthcare enterprise network, e.g., a hospital or other provider location within a multi-entity provider group.

Optionality: Optional

Format / Values

Unique identifier text string within the healthcare enterprise. May be unvalued when the audit-generating application is uniquely identified by Audit Source ID.

Rationale

This value differentiates among the sites in a multi-site enterprise health information system.

Notes

This is defined by the application that generates the audit record. It contains a unique code that identifies a business organization (owner of data) that is known to the enterprise. The value further qualifies and disambiguates the Audit Source ID. Values may vary depending on type of business. There may be levels of differentiation within the organization.

b) Audit Source ID

Description

Identifier of the source where the event originated.

Optionality: Required

Format / Values

Unique identifier text string, at least within the Audit Enterprise Site ID

Rationale

This field ties the event to a specific source system. It may be used to group events for analysis according to where the event occurred.

Notes

In some configurations, a load-balancing function distributes work among two or more duplicate servers. The values defined for this field thus may be considered as an source identifier for a group of servers rather than a specific source system.

c) Audit Source Type Code

Description

Code specifying the type of source where event originated.

Optionality: Optional

Format / Values

Coded-value enumeration, optionally defined by system implementers or as a reference to a standard vocabulary. Unless defined or referenced, the default values for the "code" attribute are:

<i>Value</i>	<i>Meaning</i>
1	End-user interface
2	Data acquisition device or instrument
3	Web server process tier in a multi-tier system
4	Application server process tier in a multi-tier system

5	Database server process tier in a multi-tier system
6	Security server, e.g., a domain controller
7	ISO level 1-3 network component
8	ISO level 4-6 operating software
9	External source, other or unknown type

For implementation defined codes or references to standards, the XML schema defines these optional attributes:

<i>Attribute</i>	<i>Value</i>
CodeSystem	OID reference
CodeSystemName	Name of the coding system; strongly recommended to be valued for locally-defined code-sets
DisplayName	The value to be used in displays and reports
OriginalText	Input value that was translated to the code

Since audit sources may be categorized in more than one way, there may be multiple values specified.

Rationale

This field indicates which type of source is identified by the Audit Source ID. It is an optional value that may be used to group events for analysis according to the type of source where the event occurred.

5. Participant Object Identification

The following data assist the auditing process by indicating specific instances of data or objects that have been accessed. These data are required unless the values for Event Identification, Active Participant Identification, and Audit Source Identification are sufficient to document the entire auditable event. Production of audit records containing these data may be enabled or suppressed, as determined by healthcare organization policy and regulatory requirements.

Because events may have more than one participant object, this group can be a repeating set of values. For example, depending on institutional policies and implementation choices:

D.3.6 Third revision of the WP3 framework

- Two participant object value-sets can be used to identify access to patient data by medical record number plus the specific health care encounter or episode for the patient.
- A patient participant and his authorized representative may be identified concurrently.
- An attending physician and consulting referrals may be identified concurrently.
- All patients identified on a worklist may be identified.
- For radiological studies, a set of related participant objects identified by accession number or study number, may be identified.

Note, though, that each audit message documents only a single usage instance of such participant object relationships and does not serve to document all relationships that may be present or possible.

a) Participant Object Type Code

Description

Code for the participant object type being audited. This value is distinct from the user's role or any user relationship to the participant object.

Optionality: Optional

Format / Values

Enumeration:

<i>Value</i>	<i>Meaning</i>
1	Person
2	System Object
3	Organization
4	Other

Rationale

To describe the object being acted upon. In addition to queries on the subject of the action in an auditable event, it is also important to be able to query on the object type for the action.

b) Participant Object Type Code Role

Description

Code representing the functional application role of Participant Object being audited

Optionality: Optional

Format / Values

Enumeration, specific to Participant Object Type Code:

<i>Value</i>	<i>Meaning</i>	<i>Participant Object Type Codes</i>
1	Patient	1 - Person
2	Location	3 - Organization
3	Report	2 - System Object
4	Resource	1 – Person 3 - Organization
5	Master file	2 - System Object
6	User	1 - Person 2 - System Object (non-human user)
7	List	2 - System Object
8	Doctor	1 - Person
9	Subscriber	3 - Organization
10	Guarantor	1 - Person 3 - Organization
11	Security User Entity	1 - Person 2 - System Object
12	Security User Group	2 - System Object
13	Security Resource	2 - System Object
14	Security Granularity Definition	2 - System Object
15	Provider	1 - Person 3 - Organization
16	Data Destination	2 - System Object
17	Data Repository	2 - System Object
18	Schedule	2 - System Object

19	Customer	3 - Organization
20	Job	2 - System Object
21	Job Stream	2 - System Object
22	Table	2 - System Object
23	Routing Criteria	2 - System Object
24	Query	2 - System Object

A "Security Resource" is an abstract securable object, e.g., a screen, interface, document, program, etc. -- or even an audit data set or repository.

Rationale

For some detailed audit analysis it may be necessary to indicate a more granular type of participant, based on the application role it serves.

c) Participant Object Data Life Cycle

Description

Identifier for the data life-cycle stage for the participant object. This can be used to provide an audit trail for data, over time, as it passes through the system.

Optionality: Optional

Format/Values

Enumeration:

<i>Value</i>	<i>Meaning</i>
1	Origination / Creation
2	Import / Copy from original
3	Amendment
4	Verification
5	Translation
6	Access / Use
7	De-identification

8	Aggregation, summarization, derivation
9	Report
10	Export / Copy to target
11	Disclosure
12	Receipt of disclosure
13	Archiving
14	Logical deletion
15	Permanent erasure / Physical destruction

Rationale

Institutional policies for privacy and security may optionally fall under different accountability rules based on data life cycle. This provides a differentiating value for those cases.

d) Participant Object ID Type Code

Description

Describes the identifier that is contained in Participant Object ID.

Optionality: Required

Format / Values

Coded-value enumeration, specific to Participant Object Type Code, using attribute-name "code". The codes below are the default set.

<i>Value</i>	<i>Meaning</i>	<i>Participant Object Type Codes</i>
1	Medical Record Number	1 – Person
2	Patient Number	1 – Person
3	Encounter Number	1 – Person
4	Enrollee Number	1 – Person
5	Social Security Number	1 – Person
6	Account Number	1 – Person 3 - Organization

7	Guarantor Number	1 - Person 3 - Organization
8	Report Name	2 - System Object
9	Report Number	2 - System Object
10	Search Criteria	2 - System Object
11	User Identifier	1 - Person 2 - System Object
12	URI	2 - System Object

User Identifier and URI [RFC2396] text strings are intended to be used for security administration trigger events to identify the objects being acted-upon.

The codes may be the default set stated above, implementation-defined, or reference a standard vocabulary enumeration, such as HL7 version 2.4 table 207 or DICOM defined media types. For implementation defined codes or references to standards, the XML schema defines these optional attributes:

Attribute	Value
CodeSystem	OID reference
CodeSystemName	Name of the coding system; strongly recommended to be valued for locally-defined code-sets
DisplayName	The value to be used in displays and reports
OriginalText	Input value that was translated to the code

Rationale

Required to distinguish among various identifiers that may synonymously identify a participant object.

e) Participant Object Sensitivity

Description

Denotes policy-defined sensitivity for the Participant Object ID such as VIP, HIV status, mental health status, or similar topics.

Optionality: Optional

Format / Values

Values are institution- and implementation-defined text strings.

f) Participant Object ID

Description

Identifies a specific instance of the participant object.

Optionality: Required

Format / Values

Text string. Value format depends on Participant Object Type Code and the Participant Object ID Type Code.

Rationale

This field identifies a specific instance of an object, such as a patient, to detect/track privacy and security issues.

Notes

Consider this to be the primary unique identifier key for the object, so it may be a composite data field as implemented.

g) Participant Object Name

Description

An instance-specific descriptor of the Participant Object ID audited, such as a person's name.

Optionality: Optional

Format / Values

Text string

Rationale

This field may be used in a query/report to identify audit events for a specific person, e.g., where multiple synonymous Participant Object IDs (patient number, medical record number, encounter number, etc.) have been used.

h) Participant Object Query

Description

The actual query for a query-type participant object.

Optionality: Optional

Format / Values

Base 64 encoded data

Rationale

For query events it may be necessary to capture the actual query input to the query process in order to identify the specific event. Because of differences among query implementations and data encoding for them, this is a base 64 encoded data blob. It may be subsequently decoded or interpreted by downstream audit analysis processing.

i) Participant Object Detail

Description

Implementation-defined data about specific details of the object accessed or used.

Optionality: Optional

Format

Type-value pair. The "type" attribute is an implementation- defined text string. The "value" attribute is a base 64 encoded data.

Rationale

Specific details or values from the object accessed may be desired in specific auditing implementations. The type-value pair enables the use of implementation-defined and locally-extensible object type identifiers and values. For example, a clinical diagnostic object may contain multiple test results, and this element could document the type and number and type of results. Many possible data encodings are possible for this elements, so the value is a base 64 encoded data blob. It may be subsequently decoded or interpreted by downstream audit analysis processing.

4.1.6. Audit implementation in WP3 Services

WP3 Services are fully audited by the Audit Trail service described above. Listed bellow are the audit implementations for each of the WP3 service.

a) RLUS

For RLUS service the following operations are audited: Start/Stop of service, CRUD operations on medical documents and Query operations. The following RLUS service interfaces have been audited: RLUSRuntime and RLUSAdministrative.

Examples of audit messages:

Start service:

```
<?xml version="1.0" encoding="utf-16"?>

<AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EventIdentification EventOutcomeIndicator="0" EventDateTime="2008-03-
05T14:31:09.921875+02:00">
    <EventID code="110100" codeSystemName="DCM" displayName="Application
Activity"/>
    <EventTypeCode code="110120" codeSystemName="DCM"
displayName="Application Start"/>
  </EventIdentification>
  <ActiveParticipant UserIsRequestor="false"
NetworkAccessPointID="192.168.100.207" UserID="SOFTINFO\abogdan" UserName="RLUS
Server">
    <RoleIDCode code="110152" codeSystemName="DCM"
displayName="Destination"/>
  </ActiveParticipant>
  <AuditSourceIdentification AuditSourceID="RLUS">
    <AuditSourceTypeCode code="4" codeSystemName="InfoWorld"
displayName="4"/>
  </AuditSourceIdentification>
</AuditMessage>
```

CRUD on documents

```
<?xml version="1.0" encoding="utf-16"?>
<AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EventIdentification EventOutcomeIndicator="0" EventDateTime="2008-03-
05T17:20:08.46875+02:00" EventActionCode="C">
    <EventID code="IRLUSRuntime.CreateResource" codeSystemName="InfoWorld"
displayName="IRLUSRuntime.CreateResource"/>
    <EventTypeCode code="IRLUSRuntimeActivity" codeSystemName="InfoWorld"
displayName="Create Resource"/>
  </EventIdentification>
  <ActiveParticipant NetworkAccessPointID="127.0.0.1:2675" UserID="SOFTINFO\abogdan">
    <RoleIDCode code="110153" codeSystemName="DCM" displayName="Source"/>
  </ActiveParticipant>
  <ActiveParticipant UserIsRequestor="false" NetworkAccessPointID="192.168.100.207"
UserID="SOFTINFO\abogdan" UserName="RLUS Server">
    <RoleIDCode code="110152" codeSystemName="DCM" displayName="Destination"/>
  </ActiveParticipant>
  <AuditSourceIdentification AuditSourceID="192.168.100.207"/>
  <ParticipantObjectIdentification ParticipantObjectID="12345">
    <ParticipantObjectIDTypeCode code="1" codeSystemName="InfoWorld"
displayName="1"/>
  </ParticipantObjectIdentification>
  <ParticipantObjectIdentification>
    <ParticipantObjectIDTypeCode code="urn:uuid:a54d6aa5-d40d-43f9-88c5-b4633d873bdd"
codeSystemName="InfoWorld" displayName="urn:uuid:a54d6aa5-d40d-43f9-88c5-
b4633d873bdd"/>
  </ParticipantObjectIdentification>
</AuditMessage>
```

Query

b) MPI

```
<?xml version="1.0" encoding="utf-8"?>
<AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EventIdentification EventOutcomeIndicator="0" EventDateTime="2008-03-
04T07:40:21.4792319Z" EventActionCode="E">
    <EventID code="110100" codeSystemName="DCM" displayName="Application Activity" />
    <EventTypeCode code="110121" codeSystemName="DCM" displayName="Application Stop"
/>
  </EventIdentification>
  <ActiveParticipant UserIsRequestor="true" NetworkAccessPointID="192.168.100.197"
NetworkAccessPointTypeCode="2" UserID="softinfo\marulisp" />
  <AuditSourceIdentification AuditSourceID="EIS">
    <AuditSourceTypeCode code="4" codeSystemName="InfoWorld" displayName="4" />
  </AuditSourceIdentification>
</AuditMessage>
```

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EventIdentification EventOutcomeIndicator="0" EventDateTime="2008-03-
04T08:57:32.2813658Z" EventActionCode="C">
    <EventID code="EISEntityActivity" codeSystemName="InfoWorld"
displayName="EISEntityActivity" />
    <EventTypeCode code="IEntityManagement.CreateEntity" codeSystemName="InfoWorld"
displayName="IEntityManagement.CreateEntity" />
  </EventIdentification>
  <ActiveParticipant UserIsRequestor="true" NetworkAccessPointID="127.0.0.1"
NetworkAccessPointTypeCode="2" UserID="SOFTINFO\marulisp" />
  <AuditSourceIdentification AuditSourceID="EIS">
    <AuditSourceTypeCode code="4" codeSystemName="InfoWorld" displayName="4" />
  </AuditSourceIdentification>
  <ParticipantObjectIdentification ParticipantObjectDataLifeCycle="1"
ParticipantObjectSensitivity="normal" ParticipantObjectID="1773be3e-bb08-44bf-9885-
9397d2640377">
    <ParticipantObjectIDTypeCode code="EntityIdentifier" codeSystemName="InfoWorld"
displayName="EntityIdentifier" />
    <ParticipantObjectName>EIS EIS TEST </ParticipantObjectName>
    <ParticipantObjectDetail type="EntityTypeIdentifier"
value="SEw3LVJJTS1WMy1QZXJzb24=" />
    <ParticipantObjectDetail type="DomainIdentifier" value="SW5mb1dvcmxk" />
    <ParticipantObjectDetail type="PAT" value="UEFU" />
  </ParticipantObjectIdentification>
</AuditMessage>
```

Query

```
<?xml version="1.0" encoding="utf-8"?>
<AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EventIdentification EventOutcomeIndicator="0" EventDateTime="2008-03-
04T09:06:44.6785823Z" EventActionCode="E">
    <EventID code="110112" codeSystemName="DCM" displayName="Query" />
    <EventTypeCode code="IQueryFunctions.FindEntitiesByTrait"
codeSystemName="InfoWorld" displayName="IQueryFunctions.FindEntitiesByTrait" />
  </EventIdentification>
  <ActiveParticipant UserIsRequestor="true" NetworkAccessPointID="127.0.0.1"
NetworkAccessPointTypeCode="2" UserID="SOFTINFO\marulisp" />
  <AuditSourceIdentification AuditSourceID="EIS">
    <AuditSourceTypeCode code="4" codeSystemName="InfoWorld" displayName="4" />
  </AuditSourceIdentification>
  <ParticipantObjectIdentification ParticipantObjectSensitivity="normal"
ParticipantObjectID="IQueryFunctions.FindEntitiesByTrait">
    <ParticipantObjectIDTypeCode code="10" displayName="Search Criteria" />
    <ParticipantObjectDetail type="MatchingAlgorithm"
value="SW5mb1dvcmxkLkhMny5JVFMuRXh0ZW5zaW9uLkVJUy5NYXRjaGluZ0FsZ29yaXRobVN0cmLuZw=="
/>
    <ParticipantObjectDetail type="InfoWorld.HL7.ITS.ST"
value="SW5mb1dvcmxkLkhMny5JVFMuU1Q=" />
  </ParticipantObjectIdentification>
</AuditMessage>
```

4.2. RLUS WS-Eventing

4.2.1. Architecture overview

WS-Eventing describes a protocol that allows Web services to subscribe to or accept subscriptions for event notification messages. A mechanism for registering interest is needed because the set of Web services interested in receiving such messages is often unknown in advance or will change over time. This specification defines a protocol for one Web service (called a "subscriber") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications" or "event messages"). The subscriber may manage the subscription by interacting with a Web service (called the "subscription manager") designated by the event source.

To improve robustness, a subscription may be leased by an event source to a subscriber, and the subscription expires over time. The subscription manager provides the ability for the subscriber to renew or cancel the subscription before it expires.

There are many mechanisms by which event sources may deliver events to event sinks. This specification provides an extensible way for subscribers to identify the delivery mechanism they prefer. While asynchronous, pushed delivery is defined here, the intent is that there should be no limitation or restriction on the delivery mechanisms capable of being supported by this specification.

A WS-Eventing implementation was developed for ReMINE platform to use it for data mining and knowledge inference system. In this deliverable, ws-eventing as defined by W3C is detailed and also the implementation for RLUS. Usage of the RLUS WS-Eventing is briefly described in deliverable D3.8 – 3rd revision of the Data Mining and Knowledge Inference module.

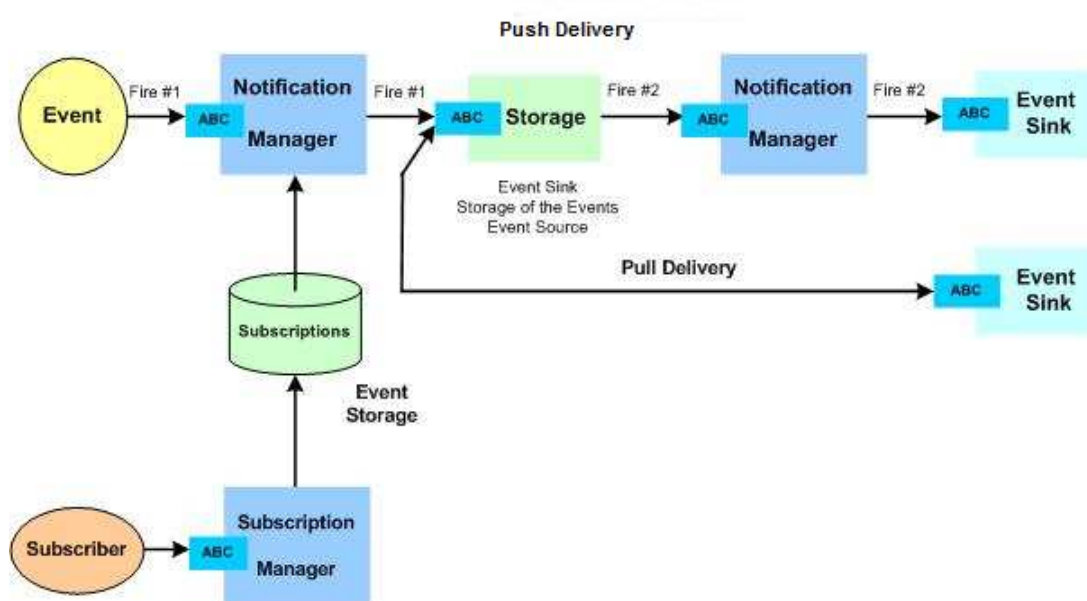


Figure 5 – WS-Eventing architecture

Delivery modes

While the general pattern of asynchronous, event-based messages is extremely common, different applications often require different event message delivery mechanisms. For instance, in some cases a

simple asynchronous message is optimal, while other situations may work better if the event consumer can poll for event messages in order to control the flow and timing of message arrival. Some consumers will require event messages to be wrapped in a standard "event" SOAP envelope, while others will prefer messages to be delivered unwrapped. Some consumers may require event messages to be delivered reliably, while others may be willing to accept best-effort event delivery.

In order to support this broad variety of event delivery requirements, this specification introduces an abstraction called a Delivery Mode. This concept is used as an extension point, so that event sources and event consumers may freely create new delivery mechanisms that are tailored to their specific requirements. This specification provides a minimal amount of support for delivery mode negotiation by allowing an event source to provide a list of supported delivery modes in response to a subscription request specifying a delivery mode it does not support.

WS-Eventing standard outlines only the Push model for notifications. This delivery is based on the asynchronous pushing (routing) the notification message to the Event Sink in the fire & forget manner and the pull delivery mode developed for ReMINE RLUS. So the following modes were developed:

- **pull**, where the event sink is responsible for polling the event source periodically and pulling notifications from the endpoint given by Event Source in the subscribe operation
- **pushWithAck**, where the event source sends a notification message and waits for its acknowledgement. Acknowledgement means that the message was received and processed successfully by the subscriber

4.2.2. Service contracts

The WS – Eventing implementation can be seen as a service that will be coupled with existing WP3 services to as a new endpoint within the same service host. The following service contracts explain the methods that this new service exposes within RLUS service.

WS – Eventing exposes two main interfaces as service contracts:

- **ISubscriptionAdministration**: Provides operations for the manipulation of subscriptions
- **IWSEventing**: Blank interface used to create a dynamic endpoint from which to receive the fired subscriptions
- **IWSEventingDispatcher**: Dispatches notifications (ws-eventing messages)
- **IMessageAdministration**: Administration for existing messages – administrative purpose only

4.2.3. Detailed functional model on each interface

Subscription Administration (ISubscriptionAdministration)

- Subscribe

Business friendly name	Create subscription
-------------------------------	---------------------

Signature	<code>string Subscribe(Subscription subscription);</code>
Description	Creates a subscription for a client to receive messages
Inputs	The subscription to be created
Outputs	Subscription identifier
Exceptions	-

- ModifySubscription

Business friendly name	Modify subscription
Signature	<code>void ModifySubscription(string id, Subscription subscription);</code>
Description	Allows for modifications to a subscription
Inputs	id – id of an old subscription subscription – the new subscription
Outputs	-
Exceptions	-

- Unsubscribe

Business friendly name	Delete a subscription
Signature	<code>void Unsubscribe(string id);</code>
Description	Allows the deletion of a subscription
Inputs	id – id of an old subscription
Outputs	-

- ListSubscriptions

Business friendly name	List subscriptions
-------------------------------	--------------------

Signature	<code>IList<Subscription> ListSubscriptions();</code>
Description	Gets all subscriptions
Inputs	-
Outputs	A list of all existing subscriptions

Notification dispatcher (IWSEventingDispatcher)

- Notification

Business friendly name	Dispatch notification
Signature	<code>void Notification(Message message);</code>
Description	Receives a message containing a notification and forwards it to matching subscriptions
Inputs	The message to dispatch
Outputs	-

Main WS-Eventing (IWSEventing)

This interface exposes no methods, it is used to create endpoints at runtime, endpoints that will accept a ws-eventing dispatched messages. This interface starts a service at the subscriber location, to listen or ask for messages from an event source.

Message administration (IMessageAdministration)

- GetAllMessages

Business friendly name	Get all messages
Signature	<code>IList<PersistedMessage> GetAllMessages();</code>
Description	Gets all messages
Inputs	-

Outputs	List of Messages
----------------	------------------

- DeleteMessage

Business friendly name	Delete a message
Signature	<code>void DeleteMessage(int messageId);</code>
Description	Allows to delete a message
Inputs	Message Identifier
Outputs	-

- PurgeMessage

Business friendly name	Delete all messages
Signature	<code>void PurgeMessages(string clientEndpoint);</code>
Description	Deletes all messages for a client (if specified), or all messages
Inputs	Client Endpoint (Optional)
Outputs	-

4.2.4. Data contracts

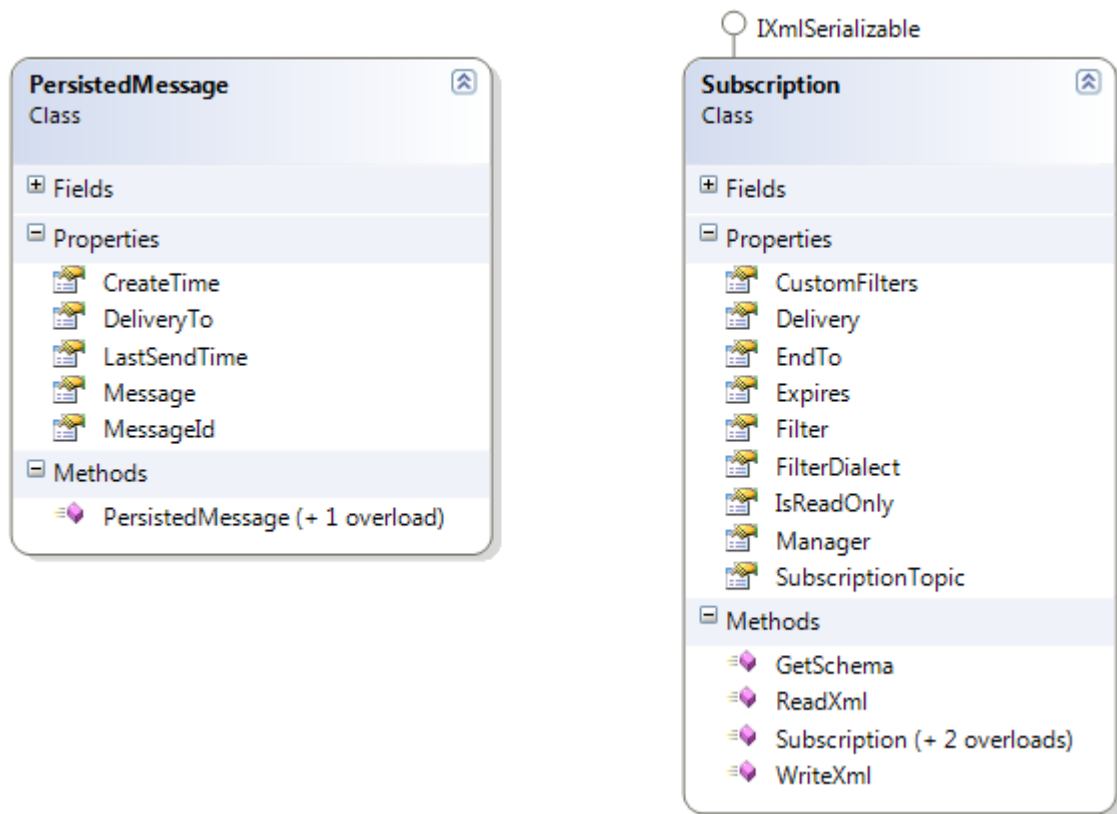


Figure 6 – WS-Eventing architecture

The main classes used for message dispatching can be seen in the data contract figure above. The Eventing message used is the same as the one proposed in the .Net framework for Web Service communication – from WCF (`System.ServiceModel.Channel.Message`).

PersistedMessage – class used to store a ws Message on the database. When fire subscription is called, a persistedMessage is created and stored in the database of the event source service. WS-Eventing runs a permanent background thread that checks the messages in the database. One thread for each of the subscribers then sends the message to the subscriber (in the case of PUSH message), in the case of PULL, a thread will be awake when the subscriber asks for messages.

Subscription – class used for administrative purposes to create a subscription with a ws-eventing ready service like RLUS. Subscription are stored in the service database in the eventing tables. (wsev schema).

Details for the fields of these classes can be found in the database layer – there is an one to one connection between the data contracts and the database layer.

4.2.5. Database layer

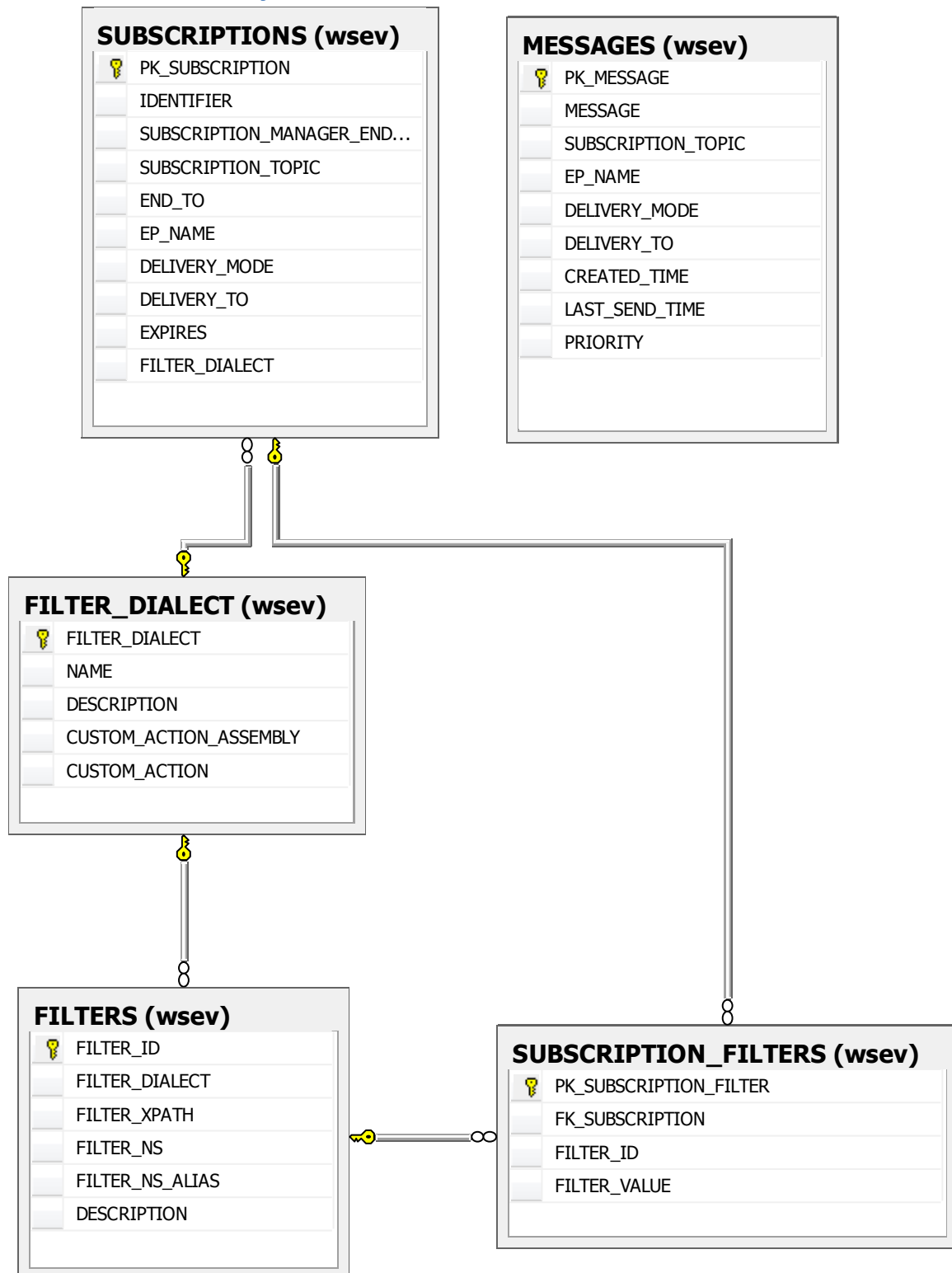


Figure 7 – WS-Eventing database structure

MESSAGES - table used to stored messages that will be sent or messages that were not sent.

Column	Data Type	Description
--------	-----------	-------------

PK_MESSAGE	int	Primary key
MESSAGE	xml	The XML serialized message to send
SUBSCRIPTION_TOPIC	nvarchar(50)	Obsolete
EP_NAME	nvarchar(250)	Endpoint name (friendly name)
DELIVERY_MODE	nvarchar(MAX)	Delivery mode (push/pull)
DELIVERY_TO	nvarchar(250)	Endpoint where to deliver the message
CREATED_TIME	datetime	Time when the message was first created
LAST_SEND_TIME	datetime	The last date/time when an attempt to send the message occurred
PRIORITY	tinyint	Priority for the messages. (0 = highest priority)

SUBSCRIPTIONS - table used to store the available subscriptions (the “link” between event source web service and subscribers)

Column	Data Type	Description
PK_SUBSCRIPTION	int	Primary key
IDENTIFIER	nvarchar(100)	GUID used to identify from UI applications this subscription
SUBSCRIPTION_MANAGER_ENDPOINT	nvarchar(MAX)	URI from where the subscription was created
SUBSCRIPTION_TOPIC	nvarchar(50)	Subscription small description
END_TO	nvarchar(250)	Endpoint where to send the message if it expires
EP_NAME	nvarchar(250)	Obsolete
DELIVERY_MODE	nvarchar(MAX)	Push/Pull delivery mode Ex: http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push
DELIVERY_TO	nvarchar(250)	Endpoint where to send the message
EXPIRES	datetime	When the message expires

FILTER_DIALECT	nvarchar(50)	What type of filter dialect to use (XPath by default)
----------------	--------------	---

FILTER_DIALECT -used to store dialects. Implicit it is XPATH, but custom dialects might exist

Column	Data Type	Description
FILTER_DIALECT	nvarchar(50)	Primary key
NAME	nvarchar(100)	Dialect friendly name
DESCRIPTION	nvarchar(250)	Explicit description of the custom dialect
CUSTOM_ACTION_ASSEMBLY	nvarchar(250)	.Net assembly containing a custom dialect
CUSTOM_ACTION	nvarchar(250)	Method in the assembly that returns a custom dialect

FILTERS - used to store existing filters

Column	Data Type	Description
FILTER_ID	int	Primary key
FILTER_DIALECT	nvarchar(50)	Foreign key to the dialect used
FILTER_XPATH	nvarchar(MAX)	In case the dialect is XPATH, the XPATH value is stored here. This XPATH value is applied on the message to extract values to be tested against the filter to decide whether the message should be send
FILTER_NS	nvarchar(250)	Namespace used in the filter
FILTER_NS_ALIAS	nvarchar(250)	Prefix for the namespace used in the filter
DESCRIPTION	nvarchar(250)	Filter description

SUBSCRIPTION_FILTERS – link between FILTERS and SUBSCRIPTION for an n..m cardinality. Contains the filter value as well

Column	Data Type	Description
PK_SUBSCRIPTION_FILTER	int	Primary key
FK_SUBSCRIPTION	int	Foreign key to the subscription
FILTER_ID	int	Foreign key to the filter
FILTER_VALUE	nvarchar(100)	The value checked for the filter. If the current subscription's filter matches this value, the message will be send.

RLUS sends notifications from all of the methods used in CRUD for resources and entries. The notification types that RLUS sends are:

```
public enum NotifyOperation
{
    Nothing,                //empty notification
    CreateRLUSEntry,        //RLUSRuntime - entry was created
    UpdateRLUSEntry,        //RLUSRuntime - entry was updated
    DeleteRLUSEntry,        //RLUSRuntime - entry was deleted
    CreateResource,         //RLUSAdministrative - resource was created
    UpdateResource,         //RLUSAdministrative - resource was updated
    DeleteResource,         //RLUSAdministrative - resource was deleted
}
```

The message sent contains the entire RLUS DataContract Entry and Resource as described in deliverable D3.5

5. Metadatabase reasoner

Archetype Definition Language (ADL) is a formal language for expressing archetypes, which are constraint-based models of domain entities, or what some might call “structured business rules”. The openEHR Archetype Object Model describes an object model equivalent of the ADL syntax. The openEHR archetype framework is described in terms of Archetype Definitions and Principles and an Archetype System.

Usage of ontology and reasoner can be achieved through an ADL parser. As ADL will be in charge of binding between data structure (HL7 V3 CDA) while template ADL (tADL) represents a semantic signifier and will be in charge of semantic data integration of the incoming RAPS entry in RLUS.

The metadatabase reasoner consists of a library that will be part of the RLUS service. It uses ADL files and tADL files (based on the ADL file) that will be stored in RLUS as template entries to check the

validity of all the RAPS entries RLUS is intended to accept (when inserting/updating resources). For instance, we have a new RAPS entry to store in RLUS (either CDA – adverse events and situations or non-CDA – processes). This RLUS entry will contain the template id that is required to check the semantic integrity of the entry. RLUS will access the reasoner service and wait for an answer regarding the validity of the new entry. Once received the answer, it will notify the caller in case of failure.

The final version for the ADL parser and templates has been completed in the form of a RLUS library for the parser (the reasoner) and custom templates based on the ADL language and HL7 Templates specifications in the form of a SIMP template. Description for the SIMP template package will follow with also samples for the templates used in ReMINE platform scenario.

A SIMP/SMP package (simple data template) is used to describe sections/or the header from CDA files. It uses an extension of the HL7 Templates to define its language.

(<http://www.hl7.org/v3ballot/html/infrastructure/templates/templates.htm>)

In a SIMP package using XML language parts of a CDA are described. Multiple SIMP packages are uploaded in RLUS as a set of sub templates and form the semantic signifiers used for ReMINE. After creating the source for a SIMP package, the package must be compiled where it generates C# classes and an on the fly C# project. This is built on the .net framework (and optionally on the Silverlight platform – if any RLUS client that is web based wants to validate the CDAs on his premises) into assemblies that RLUS will use to check the integrity for incoming CDA files from DEMS.

There are two types of SIMP packages – the one that defines the CDA header and that define a certain CDA section. While the CDA header has a relatively fixed structure the other sections are strictly defined by the CDA implementation guide developed for each of the pilots.

To easily create the SIMP packages a SIMP designer was created by creating a simple AddIn for the open-source SharpDevelop IDE to create the XML file – the source file for SIMP packages:

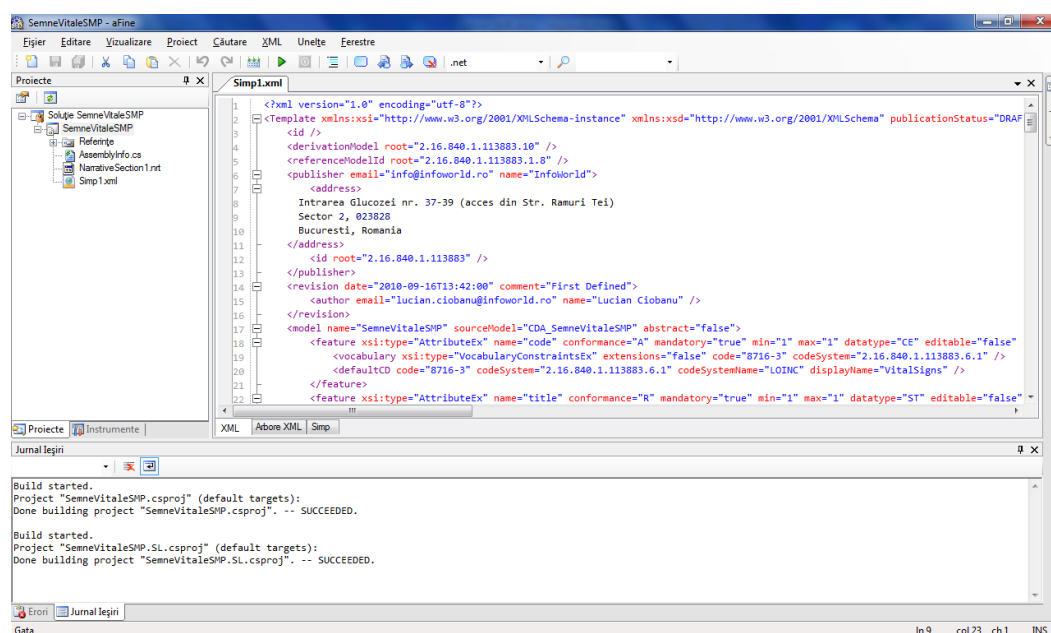


Figure 8 – SIMP Designer for VitalSigns CDA section

SIMP Designer works with the HL7 Data Types as described in the below table:

Data type	Name	Description
AD	Postal Address	Mailing and home or office addresses. A sequence of address parts, such as street or post office Box, city, postal code, country, etc.
ANY	Anything	Defines the basic properties of every data value. This is an abstract type, meaning that no value can be just a data value without belonging to any concrete type. Every concrete type is a specialization of this general abstract DataValue type.
CD	Concept description	A concept descriptor represents any kind of concept usually by giving a code defined in a code system. A concept descriptor can contain the original text or phrase that served as the basis of the coding and one or more translations into different coding systems. A concept descriptor can also contain qualifiers to describe, e.g., the concept of a "left foot" as a postcoordinated term built from the primary code "FOOT" and the qualifier "LEFT". In cases of an exceptional value, the concept descriptor need not contain a code but only the original text describing that concept.
CE	Coded with Equivalents	Coded data that consists of a coded value (CV) and, optionally, coded value(s) from other coding systems that identify the same concept. Used when alternative codes may exist.
CS	Simple coded values	Coded data in its simplest form, where only the code is not predetermined. The code system and code system version are fixed by the context in which the CS value occurs. CS is used for coded attributes that have a single HL7-defined value set.
CV	Coded Value	Coded data, specifying only a code, code system, and optionally display name and original text. Used only as the data type for other data types' properties.
ED	Encapsulated data	Data that is primarily intended for human interpretation or for further machine processing outside the scope of HL7. This includes unformatted or formatted written language, multimedia data, or structured information in as defined by a different standard (e.g., XML-signatures.)
EN	Entity Name	A name for a person, organization, place or thing. A sequence of name parts, such as given name or family name, prefix, suffix, etc
II	Instance identifier	An identifier that uniquely identifies a thing or object. Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance

		identifiers are defined based on ISO object identifiers.
IVL	Interval	A set of consecutive values of an ordered base data type. Any ordered type can be the basis of an interval; it does not matter whether the base type is discrete or continuous. If the base data type is only partially ordered, all elements of the interval must be elements of a totally ordered subset of the partially ordered data type.
ON	Organization Name	An EN used when the named Entity is an Organization. A sequence of name parts.
PN	Person Name	An EN used when the named Entity is a Person. A sequence of name parts, such as given name or family name, prefix, suffix, etc. A name part is a restriction of entity name part that only allows those entity name parts qualifiers applicable to person names. Since the structure of entity name is mostly determined by the requirements of person name, the restriction is very minor.
PQ	Physical Quantity	A dimensioned quantity expressing the result of measuring.
RTO	Property of ratio	The quantity that is being divided in the ratio. The default is the integer number 1 (one.)
SC	Character string with code	A character string that optionally may have a code attached. The text must always be present if a code is present. The code is often a local code.
ST	Character string	The character string data type stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) Used for names, symbols, and formal expressions.
TEL	Telecommunication address	A telephone number (voice or fax), e-mail address, or other locator for a resource mediated by telecommunication equipment. The address is specified as a Universal Resource Locator (URL) qualified by time specification and use codes that help in deciding which address to use for a given time and purpose.
TS	Timestamp	A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression. Semantically, however, time is independent from calendars and best described by its relationship to elapsed time (measured as a physical quantity in the dimension of time.) A point in time plus an elapsed time yields another point in time. Inversely, a point in time minus another point in time yields an elapsed time.

SIMP package structure

A SIMP Package (windows file with SMP extension) contains several components archived in a ZIP archive:

- Two assemblies in the root folder: the semantic template compiled on the .net platform and optionally on the Silverlight platform. These files (DLL) are versioned and not .net signed
- [Content_Types].xml and _rels\.rels – files used for package metadata
- Src folder contains the actual sources from this package:
 - o Simp1.xml – the template XML (main source)
 - o NarrativeSection1.nrt – XML file used to automatically generate the narrative for this section (only if required by the CDA implementation guide)
 - o .SLN and .SMPPROJ files – the .net solution and c# project file used in the compilation of the template
 - o AssemblyInfo.cs – auto incremented version file for the whole package. Each rebuild of the package changes the version of the assemblies.

Example for Vital Signs SIMP template package:

VitalSignsSMP\VitalSignsSMP.dll

VitalSignsSMP\VitalSignsSMP.SL.dll

VitalSignsSMP\Src

VitalSignsSMP\[Content_Types].xml

VitalSignsSMP_rels

VitalSignsSMP\Src\AssemblyInfo.cs

VitalSignsSMP\Src\NarrativeSection1.nrt

VitalSignsSMP\Src\VitalSignsSMP.sln

VitalSignsSMP\Src\VitalSignsSMP.smpproj

VitalSignsSMP\Src\Simp1.xml

VitalSignsSMP_rels\.rels

SIMP template structure

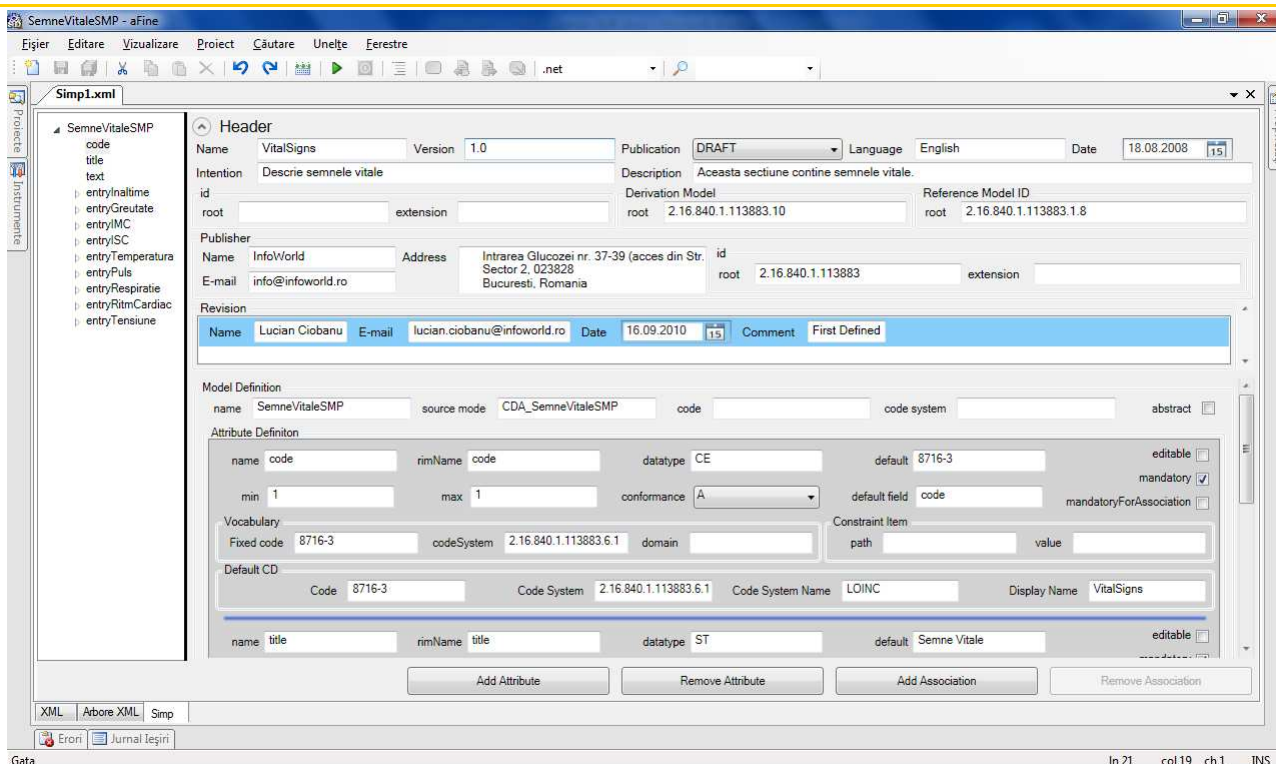


Figure 9 – SIMP Designer for VitalSigns – SIMP Header

The header part of a SIMP template contains information about the metadata for the template. These information can be seen in the XML below:

```
<?xml version="1.0" encoding="utf-8"?>
<Template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  publicationStatus="DRAFT"
  publicationStatusChangeDate="2008-08-18T12:40:00+03:00"
  version="1.0" intention="Descrie semnele vitale"
  name="VitalSigns" description="Aceasta sectiune contine semnele vitale."
  descriptionLanguage="English" format="HL7.Template.V1"
  xmlns="uri:hl7.org::template">

  <id />
  <derivationModel root="2.16.840.1.113883.10" />
  <referenceModelId root="2.16.840.1.113883.1.8" />
  <publisher email="info@infoworld.ro" name="InfoWorld">
    <address>
      Intrarea Glucozei nr. 37-39 (acces din Str. Ramuri Tei)
      Sector 2, 023828
      Bucuresti, Romania
    </address>
    <id root="2.16.840.1.113883" />
  </publisher>
  <revision date="2010-09-16T13:42:00" comment="First Defined">
    <author email="lucian.ciobanu@infoworld.ro" name="Lucian Ciobanu" />
  </revision>
  ...
  ...
  ...
</Template>
```


Item	Description
Template.Name	Template name. Inherited HL7 Templates. Informatory
Template.Version	Template version. Numeric format X.X. Inherited HL7 Templates. Informatory
Template.Publication	Publishing template status. Inherited HL7 Templates. Informatory
Template.Lanague	The language used in the template. Inherited HL7 Templates. Informatory
Template.Date	Template creation date. Inherited HL7 Templates. Informatory
Template.Intention	Free-text for the intent and purpose of this template. Inherited HL7 Templates. Informatory
Template.Description	Free-text description of the template. Inherited HL7 Templates. Informatory
Template. Id	An II data type (root/extension) for the Id of the used template. It translates in the CDA section in section.templateId
Template.DerivationModel	Globally unique indentifier for CIM(Constrained Information Model). Inherited HL7 Templates. Informatory
Template.ReferenceModelId	Globally unique indentifier for the reference model derived from the model. For Hl7 the model is RIM (Reference Information Model). Inherited HL7 Templates. Informatory
Template.Publisher	Information about the author of the Template. May contain name, email, adress and ID (II datatype). Inherited HL7 Templates. Informatory
Template.Revision	Information about „revision history”. Last revision, what has changed, date and author. Inherited HL7 Templates. Informatory

These items must be filled in the header of any SIMP package and they are seen as metadata for that SIMP package. The Template.Id is very important - this is translated into the template ID on the section and must comply with the CDA implementation guide for that section.

SIMP actual content is found in the XML node „Model”. As attributes to this node we have:

Item	Description
Template.Model.name	Model name. SIMP default name. Used as namespace when generating .NET classes
Template.Model.sourceModel	Source model name. Default is „CDA_” + SIMP package name. Used as namespace when generating .NET classes
Template.Model.abstract	False if the model contains attributes and/or association –default value.

Template.Model.sectionPath	Path to identify the existing section. It is edited from the designer in the Model Definition\code and code system. For code="48765-2" and codeSystem = „2.16.840.1.113883.6.1" in XML is translated into: <code>sectionPath="/ClinicalDocument/component/structuredBody/component[section/code/@code='48765-2' and section/code/@codeSystem='2.16.840.1.113883.6.1']/section"</code>
----------------------------	---

To add attributes/associations to the current element use the buttons „Add Attribute" / „Add Association" in the visual editor of SIMPs, having selected on the left site, in the tree diagram, the desired item:

- On a model we can add attributes and/or associations
- Nothing can be added on an attribute
- On associations we can add attributes and/or associations

When selecting an attribute or association it can be deleted using the „Remove Attribute" / „Remove Association" buttons, thus removing it from the SIMP's XML.

Attributes

A Model consists of a tree with nodes made of Attributes and associations. According to HL7 Templates, a Model contains Artifacts which specialize in Features (extended artefacts) which is a base class for Attribute and Association. We use AttributeEx and AssociationEx as extensions of the homologous HL7 Templates.

For AttributeEx we have the following items:

Item	Description
name	Attribute name. The names begin with lower case for attributes and generate a property in the parent or in .net classes (with get and set)
rimName	String for the name in RIM (ranking of classes in HL7 CDA - ClinicalDocument). Attention: it represents the hierarchy from the HL7CDA assembly not from a CDA's XML.
conformance	Enum to determine if the feature is (A)llowed, (R)equired or (NP) Not Permitted. Informatory
mandatory	Boolean for mandatory attribute. If it is true and there is a template field that must be completed and was not specified, when saving a validation error will occur because this had to be completed.
min/max	How relevant is an attribute. Min is informative. If Max is more than 1 (<code>max="2147483647"</code> - <code>maxInt</code>) when generating .net classes it will generate the property as <i>Collection</i> (<code>ObservableCollection<T></code>)
dataType	The datatype for this rimName. (Ex. For „Attribute Title from the model becomes a property" it is ST)
default/defaultField	The option to set default values is a field from this item (used as a datatype). Ex: if we want to set default for title as „Vital signs": <code>default="Vital Signs" defaultField="text"</code> – on the ST datatype from the field the value „Vital Signs" will be placed. This will translate in the generated .net classes as: <pre>this._title = new ST(); this._title.text = "Vital Signs";</pre>
editable	Boolean. True if at SIMP loading in design tools this will appear in the tree. If it's true, it will also generate the set of columns in the reports DataSource and LINQ query for this field. False otherwise
mandatoryForAssociation	Boolean. True if the item is mandatory for the association to which it belongs. If it's true and the item has not been completed, association is invalid at translation – it is regarded as a key for the association.

vocabulary.code	For CD data type attributes (CE, CV etc.) in the Vocabulary node we can put constraints on code/codeSystem. In vocabulary.code the code for the CD is stored. Attribution of another code in this attribute will throw an exception. (This constrain is reflected in the .net class as property in Setter by checking the passed value. If it is different an ArgumentException is thrown)
vocabulary.codeSystem	The same as vocabulary.code but checks the codeSystem from CD
vocabulary.domain	This can appear independently from code or codeSystem and represents the value set that will be used for the CD.
constraintItem	For any attribute (and association) a constraint can be created to be used in CDA-SIMP translation. If there are two or more attributes or associations on the same rimName but they are different inside the attribute(association), in order to know which is loaded to which, a constraintItem with path (path to the different element) and value (the value of the element) can be used.
constraintItem.path	Path (composed rimName) to obtain the element.
constraintItem.value	The value from the constraintItem to be verified.
defaultCD	For CD data types all four components with default values can be added (code / codeSystem / DisplayName / codeSystemName). They are transposed in the generated .net code by assigning in the parent constructor (the model or an association)

Examples:

- 1) If we have this attribute directly on the model:

```
<feature xsi:type="AttributeEx" name="title" conformance="R" mandatory="true" min="1"
max="1" datatype="ST"
editable="false" rimName="title" default="Vital Signs" defaultField="text">
</feature>
```

in cda it will become Section.Title with the value „Vital signs“, uneditable

In the generated .net code we have:

- On the parent (model) a ST property named Title is created. Because it's mandatory true, it cannot have null values:

```
[DefaultAttribute("text", "Vital Signs"),
Editable(false), IsMandatory,
XmlElement(ElementName="title", Order=1)]
public ST Title
{
    get
    {
        return this._title;
    }
    set
    {
        if (this._title != value)
        {
            if (value == null)
            {
                throw new ArgumentNullException("Title");
            }
            this._title = value;
            this.RaisePropertyChanged("Title");
        }
    }
}
```

In the parent constructor the following code is generated:

```
this._title = new ST();
this._title.text = "Vital Signs";
```

- 2) If we have a code attribute „Section/Entry/Observation“ in Vital Signs

```
<feature xsi:type="AttributeEx" name="code" conformance="A" mandatory="false" min="1"
max="1" datatype="CD">
```

```
editable="false" rimName="code">
<vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="363808001"
codeSystem="2.16.840.1.113883.6.96" />
<defaultCD code="363808001" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
displayName="Body weight measure" />
</feature>
```

In CDA it will become:

```
<component>
  <section xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3">
    ...
    <entry>
      <observation classCode="OBS" moodCode="EVN">
        <code displayName="Body weight measure" codeSystemName="SNOMED CT" code="363808001"
          codeSystem="2.16.840.1.113883.6.96" />
        ...
      </observation>
      ...
    </entry>
    ...
  </section>
</component>
```

On the parent class the following is generated:

```
[Vocabulary(""), CDValue("363808001", "2.16.840.1.113883.6.96", "SNOMED CT", "Body weight
measure", ""),
Editable(false), XmlElement(ElementName="code", Order=0)]
public CD Code
{
    get
    {
        return this._code;
    }
    set
    {
        if (this._code != value)
        {
            if (value.code != "363808001")
            {
                throw new ArgumentException("Code");
            }
            this._code = value;
            this.RaisePropertyChanged("Code");
        }
    }
}
```

And in the parent constructor:

```
this._code = new CD();
this._code.code = "363808001";
this._code.codeSystem = "2.16.840.1.113883.6.96";
this._code.codeSystemName = "SNOMED CT";
this._code.displayName = "Body weight measure";
```

3) For the „name” attribute from the association below:

```
<feature xsi:type="AssociationEx" name="authorAssignedPerson" conformance="A" mandatory="false"
min="0" max="1" rimName="assignedPerson">
  <target name="AuthorAssignedPerson" sourceModel="CDA_Header" abstract="false">
    <feature xsi:type="AttributeEx" name="name" conformance="A" mandatory="false" min="0"
max="2147483647" datatype="PN" rimName="name">
    </feature>
    <derives xref="InfoWorld.HL7.RIM.Person" />
  </target>
</feature>
```

In CDA at „assignedPerson” there will be multiple „name” nodes –CDA structure permits this:

```
<assignedPerson>
  <name>
    <family>Doe</family>
    <given>Joe</given>
  </name>
  <name>
```

```
<family>Doe the 1st</family>
<given>Joe</given>
</name>
</assignedPerson>
```

In the .net generated code, for the generated AssignedPerson class and NameList property we get:

```
[XmlRoot(Namespace = "urn:hl7-org:v3", IsNullable = true, ElementName = "AssignedPerson"),
XmlType(Namespace = "urn:hl7-org:v3", TypeName = "AssignedPerson"),
GeneratedCode("aFine Code Generator", "1.0.0000")]
public class AssignedPerson : INotifyPropertyChanged
{
    // Fields
    private ObservableCollection<PN> _nameList;
    private PropertyChangedEventHandler PropertyChanged;

    // Events
    public event PropertyChangedEventHandler PropertyChanged;

    // Methods
    public AssignedPerson();
    protected void RaisePropertyChanged(string propertyName);

    // Properties
    [XmlElement(ElementName="name", Order=0), Editable(true)]
    public ObservableCollection<PN> NameList
    {
        get
        {
            return this._nameList;
        }
        set
        {
            if (this._nameList != value)
            {
                this._nameList = value;
                this.RaisePropertyChanged("NameList");
            }
        }
    }
}
```

Because of the cardinality number 2147483647 the property isn't „Name" but „NameList". The type is no longer PN but ObservableCollection<PN>.

4) Exemple using defaultField=".", enum and functions

```
<feature xsi:type="AttributeEx" name="statusCode" conformance="A"
mandatory="false" min="0" max="1" datatype="CS" editable="false"
rimName="statusCode"
default="completed" defaultField="code">
</feature>
<feature xsi:type="AttributeEx" name="classCode" conformance="A" mandatory="true" min="0"
max="0"
datatype="ActClassObservation" editable="false" rimName="classCode"
default="COND" defaultField=".">
</feature>
<feature xsi:type="AttributeEx" name="id" conformance="R" mandatory="true" min="1" max="1"
datatype="II" rimName="id" default="FUNCTION:NewGUID" defaultField="root">
</feature>
```

The first attribute is for generating a node in the CDA XML of type `<statusCode code="completed" />`

The second is generating a classCode attribute with the value COND on the referred XML node:

```
<observation classCode="COND" />
```

The third generates in CDA an id with root which is set to a new GUID: `<id root="6AEA48F4-9488-450D-8313-88FCD3476656" />` Three other functions exist:

- FUNCTION:NewGUID: generates a new GUID
- FUNCTION: CurrentDate: generates a DateTime of form „yyyyMMddHHmmss" for the current date
- FUNCTION: CurrentDateTime: generates a DateTime of form „yyyyMMddHHmmss" for the current date and time

Associations

Associations represent Container classes for other associations or attributes. For example on a Section we have the Entry node. Section is in SIMP the Model while Entry will be an association because the Entry also contains other nodes in the CDA XML. For an association it is important to specify the type it inherits. (The Model derives from InfoWorld.HL7.CDA.Section and the Entry derives from InfoWorld.HL7.CDA.Entry).

There are some cases where we have a node in XML that at first glance appears to be the attribute. (Eg. RTO_PQ_PQ) but we want to break this attribute into several attributes, therefore it can be turned into an Association with those attributes (nominator, denominator as PQ attributes).

The association is represented in the reasoner designer by the AssociationEx class which derives from HL7 Templates Association. If only information (elements) proposed by the HL7 Templates are to be used, this can be done by modifying this SIMP XML AssociationEx in Association.

Items of an association (AssociationEx)

Item	Description
name	Association name. This name begins with a lowercase letter and will generate the .Net class corresponding to an association at build time
rimName	String for the name in RIM (ranking of classes in HL7 CDA - ClinicalDocument). Attention: it represents the hierarchy from HL7CDA assembly, not from a CDA's XML.
conformance	Enum to determine if the feature is (A)llowed, (R)equired or (NP) Not Permitted. Informatory
mandatory	Boolean for mandatory association.
min/max	How relevant is an attribute. Min is informative. If Max is greater than 1 (<code>max="2147483647"</code> - <code>maxInt</code>) when generating .net classes, it will generate the property as <i>Collection</i> (<code>ObservableCollection<AssociationName></code>)
derives	Data type of association. A form of filtering is used for HL7CDA and HL7DataTypes assemblies. Examples: <pre><derives xref="InfoWorld.HL7.CDA.Entry" /> <derives xref="InfoWorld.HL7.RIM.Observation" /></pre>
constraintItem	For any attribute (and association) a constraint can be created to be used in CDA-SIMP translation. If there are two or more attributes or associations on the same rimName but they are different inside the attribute(association), in order to know which is loaded to which, a constraintItem with path (path to the different element) and value (the value of the element).
constraintItem.path	Path (composed rimName) to obtain the item
constraintItem.value	The value of the item from constraintItem to be verified.
mandatoryForAssociation	Boolean. True if this element is required for the association to which it belongs. If True and the item was not completed, the association will not be translated – it is seen as key to the parent association. It is used in combination with other mandatoryForAssociation = true associations and a true attribute on mandatoryForAssociation.

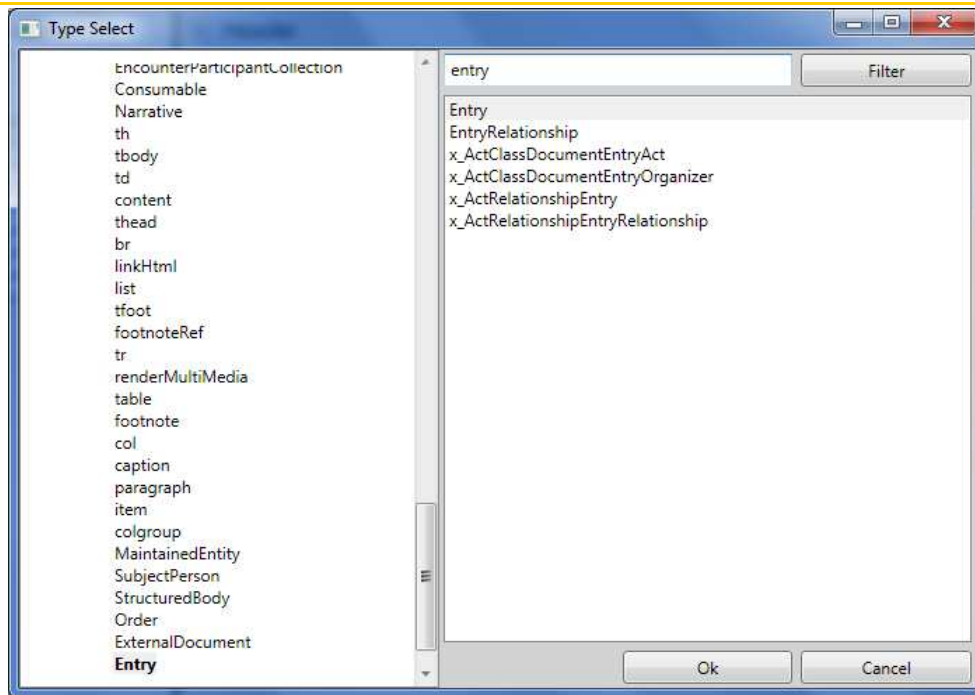


Fig. Filtering form for „Derives” for Associations

Observations:

- in a direct change in XML, the name of the feature (AssociationEx) should start with a lowercase letter
- when creating an association in the CDA only items describing the attributes and components of this association in block of the section will be saved. This is the desired behavior for a SIMP package, but if for example the modification of certain CDA components is desired, the whole structure around the component needs to be described in SIMP. Example: we want to handle the patient's name. To do this in the header type SIMP an association corresponding to the recordTarget is created. If not all attributes and associations for a maximal case are defined for the recordTarget, data will be lost upon saving the CDA. Input CDA may contain the patient address, but if the corresponding AD attribute was not defined in SIMP, the patient address will be lost upon saving the CDA.

Examples:

1) Ex1:

We have the two associations below, the height and weight from Vital Signs

```
<feature xsi:type="AssociationEx" name="entryInalttime" conformance="A"
  mandatory="false" min="1" max="1" rimName="entry">
<target xsi:type="ConstrainedClassEx" name="EntryInalttime" abstract="false">
  <feature xsi:type="AssociationEx" name="observationInalttime" conformance="A"
    mandatory="false" min="1" max="1" rimName="observation"
    mandatoryForAssociation="true">
<target xsi:type="ConstrainedClassEx" name="ObservationInalttime" abstract="false">
  <feature xsi:type="AttributeEx" name="code" conformance="A"
    mandatory="false" min="1" max="1" datatype="CD" editable="false"
    rimName="code">
<vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
  code="50373000" codeSystem="2.16.840.1.113883.6.96" />
<defaultCD code="50373000" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
  displayName="Body height measure" />
</feature>
<feature xsi:type="AttributeEx" name="statusCode" conformance="A" mandatory="false" min="0"
  max="1" datatype="CS" editable="false" rimName="statusCode"
  default="completed" defaultField="code">
</feature>
<feature xsi:type="AttributeEx" name="effectiveTime" conformance="A"
  mandatory="false" min="1" max="1" datatype="IVL_TS" rimName="effectiveTime">
</feature>
<feature xsi:type="AttributeEx" name="value" conformance="A" mandatory="false"
  min="1" max="1" datatype="PQ" rimName="value" mandatoryForAssociation="true"
  default="Cm" defaultField="unit">
</feature>
```

```
<derives xref="InfoWorld.HL7.RIM.Observation" />
</target>
</feature>
<derives xref="InfoWorld.HL7.CDA.Entry" />
</target>
</feature>
<feature xsi:type="AssociationEx" name="entryGreutate" conformance="A" mandatory="false" min="1"
        max="1" rimName="entry">
    <target xsi:type="ConstrainedClassEx" name="EntryGreutate" abstract="false">
        <feature xsi:type="AssociationEx" name="observationGreutate" conformance="A" mandatory="false"
min="1"
        max="1" rimName="observation" mandatoryForAssociation="true">
            <target xsi:type="ConstrainedClassEx" name="ObservationGreutate" abstract="false">
                <feature xsi:type="AttributeEx" name="code" conformance="A" mandatory="false" min="1"
max="1"
                datatype="CD" editable="false" rimName="code">
                    <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="363808001"
codeSystem="2.16.840.1.113883.6.96" />
                    <defaultCD code="363808001" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
displayName="Body weight measure" />
                </feature>
                <feature xsi:type="AttributeEx" name="statusCode" conformance="R" mandatory="false" min="0"
max="1" datatype="CS" editable="false" rimName="statusCode"
                default="completed" defaultField="code">
                </feature>
                <feature xsi:type="AttributeEx" name="effectiveTime" conformance="A" mandatory="false"
min="1"
                max="1" datatype="IVL_TS" rimName="effectiveTime">
                </feature>
                <feature xsi:type="AttributeEx" name="value" conformance="A" mandatory="false" min="1"
max="1"
                datatype="PQ" rimName="value" mandatoryForAssociation="true" default="Kg"
defaultField="unit">
                </feature>
            <derives xref="InfoWorld.HL7.RIM.Observation" />
        </target>
    </feature>
    <derives xref="InfoWorld.HL7.CDA.Entry" />
</target>
</feature>
```

The CDA below will be generated:

```
<entry>
  <observation classCode="OBS" moodCode="EVN">
    <code displayName="Body height measure" codeSystemName="SNOMED CT" code="50373000"
codeSystem="2.16.840.1.113883.6.96" />
    <statusCode code="completed" />
    <effectiveTime value="20101031020000.0000+03" />
    <value xsi:type="PQ" unit="Cm" value="178" />
  </observation>
</entry>
<entry>
  <observation classCode="OBS" moodCode="EVN">
    <code displayName="Body weight measure" codeSystemName="SNOMED CT" code="363808001"
codeSystem="2.16.840.1.113883.6.96" />
    <statusCode code="completed" />
    <effectiveTime value="20101031020000.0000+03" />
    <value xsi:type="PQ" unit="Kg" value="65" />
  </observation>
</entry>
```

The difference between the 2 association is that the entry.observation is different, and it acts like a key for observation. Because we have:

```
<vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
code="50373000" codeSystem="2.16.840.1.113883.6.96" />
<defaultCD code="50373000" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
displayName="Body height measure" />
```

on the first association, and

```
<vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="363808001"
codeSystem="2.16.840.1.113883.6.96" />
<defaultCD code="363808001" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"
displayName="Body weight measure" />
```


On the second association, this Vocabulary + DefaultCD functions like a constraintItem; when loading and translating from CDA -> SIMP, the above CDA will load correctly the 2 associations – first entry will be found in the association with the entry.code.code = 50373000 and second entry in the association with the 363808001 entry.code.code.

2) Ex2 – Association with maxint cardinality:

```
<feature xsi:type="AssociationEx" name="entry" conformance="R" mandatory="true" min="1"
max="2147483647"
    rimName="entry">
    <target xsi:type="ConstrainedClassEx" name="Entry" abstract="false">
        <feature xsi:type="AssociationEx" name="observation" conformance="A" mandatory="false"
min="1" max="1"
            rimName="observation">
                <target xsi:type="ConstrainedClassEx" name="Observation" abstract="false">
                    <feature xsi:type="AttributeEx" name="code" conformance="R" mandatory="true" min="1"
max="1"
                        datatype="CD" rimName="code">
                            <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
                                domain="DiagnosisTypeAssessment" codeSystem="2.16.840.1.113883.6.96"
                            />
                        />
                    </feature>
                    <derives xref="InfoWorld.HL7.RIM.Observation" />
                </target>
            </feature>
        <derives xref="InfoWorld.HL7.CDA.Entry" />
    </target>
</feature>
```

If the maximum cardinality of an association is greater than the one that generates in the parent class a property of typeObservableCollection <entry> in this case.

3) Ex3 – Creating a simple QUALIFIER:

```
<feature xsi:type="AssociationEx" name="observation" conformance="A" mandatory="false" min="1"
max="1"
    rimName="observation">
    <target xsi:type="ConstrainedClassEx" name="Observation" abstract="false">
        <feature xsi:type="AttributeEx" name="code" conformance="R" mandatory="true" min="1" max="1"
            datatype="CD" rimName="code">
                <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
                    domain="DiagnosisTypeAssessment"
                    codeSystem="2.16.840.1.113883.6.96" />
            </feature>
        <feature xsi:type="AssociationEx" name="qualifier" conformance="R" mandatory="false" min="1"
max="1"
            rimName="code.qualifier">
                <target xsi:type="ConstrainedClassEx" name="Qualifier" abstract="false">
                    <feature xsi:type="AttributeEx" name="name" conformance="A" mandatory="false" min="0"
max="1"
                        datatype="CV" editable="false" rimName="name">
                            <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="43749003"
codeSystem="2.16.840.1.113883.6.96" />
                            <defaultCD code="43749003" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED
CT"
                                displayName="Has severity" />
                        </feature>
                    <feature xsi:type="AttributeEx" name="value" conformance="A" mandatory="false" min="0"
max="1"
                        datatype="CD" rimName="value" mandatoryForAssociation="true">
                            <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" domain="DiseaseType"
codeSystem="3.3.3.3.84" />
                        </feature>
                    <derives xref="InfoWorld.HL7.ITS.CR" />
                </target>
            </feature>
        ...
    </target>
</feature>
```

In the example above, we have the Observation association which contains a code attribute. This code of CD type has a qualifier (CR with code and value). This example could translate to a CDA sequence as

```
<entry>
    <observation classCode="COND">
```

```

<code displayName="Alte boli ale intestinelor (K55-K63)" codeSystemName="ICD10AM"
code="120"
  codeSystem="2.16.840.1.113883.6.135">
    <qualifier>
      <name displayName="Has severity" codeSystemName="SNOMED CT"
      code="43749003" codeSystem="2.16.840.1.113883.6.96" />
      <value displayName="Absces ischiorectal" codeSystemName="ICD10AM"
      code="K61.3" codeSystem="3.3.3.3.84" />
    </qualifier>
  </code>
  ...
</observation>
</entry>

```

In order to model the qualifier in SIMP, a qualifier association is created, which has two attributes (name of type CV and value of type CD).

rimName for this association is "code.qualifier" (to connect with the code attribute). It also needs to be declared in SIMP immediately after the code attribute.

4) Ex4 – Creating a complex QUALIFIER:

```

<feature xsi:type="AttributeEx" name="code" conformance="A" mandatory="true" min="0" max="1"
  datatype="CE" editable="false" rimName="code">
  <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="PROV"
  codeSystem="2.16.840.1.113883.5.110" />
  <defaultCD code="PROV" codeSystem="2.16.840.1.113883.5.110" codeSystemName="RoleClass"
  displayName="healthcare provider" />
</feature>
<feature xsi:type="Association" name="specialitateMedicAtnd" conformance="R"
  mandatory="true" min="1" max="1" rimName="code.qualifier">
  <target name="SpecialitateMedicAtnd" sourceModel="CDA_Header" abstract="false">
    <feature xsi:type="AttributeEx" name="numeAtribut" conformance="A" mandatory="false"
    min="1" max="1" datatype="CV" rimName="name" default="394658006"
    defaultField="code">
      <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="394658006"
      codeSystem="2.16.840.1.113883.6.96" />
      <defaultCD code="394658006" codeSystem="2.16.840.1.113883.6.96" />
    </feature>
    <feature xsi:type="AttributeEx" name="valoareAtribut" conformance="A" mandatory="false"
    min="0" max="1" datatype="CD" rimName="value" mandatoryForAssociation="true">
      <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
      domain="SpecialitatiMedicale" codeSystem="10.10.10.10.40" />
    </feature>
    <feature xsi:type="AssociationEx" name="valoare" conformance="A" mandatory="false"
    min="0" max="1" rimName="value.qualifier">
      <target xsi:type="ConstrainedClassEx" name="Valoare" abstract="false">
        <feature xsi:type="AttributeEx" name="name" conformance="A" mandatory="false"
        min="0" max="1" datatype="CV" rimName="name">
        </feature>
        <feature xsi:type="AttributeEx" name="grad" conformance="A" mandatory="false"
        min="0" max="1" datatype="CD" rimName="value">
          <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false"
          codeSystem="10.10.10.10.41" />
        </feature>
        <derives xref="InfoWorld.HL7.ITS.CR" />
      </target>
    </feature>
    <derives xref="InfoWorld.HL7.ITS.CR" />
  </target>
</feature>
<feature xsi:type="Association" name="casaAsigurariAtnd" conformance="R" mandatory="true"
  min="1" max="1" rimName="code.qualifier">
  <target name="CasaAsigurariAtnd" sourceModel="CDA_Header" abstract="false">
    <feature xsi:type="AttributeEx" name="numeAtribut" conformance="A" mandatory="false"
    min="1" max="1" datatype="CV" rimName="name" default="UNDWRT" defaultField="code">
      <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" code="UNDWRT"
      codeSystem="2.16.840.1.113883.5.110" />
      <defaultCD code="UNDWRT" codeSystem="2.16.840.1.113883.5.110" />
    </feature>
    <feature xsi:type="AttributeEx" name="valoareAtribut" conformance="A" mandatory="false"
    min="0" max="1" datatype="CD" rimName="value" mandatoryForAssociation="true">
      <vocabulary xsi:type="VocabularyConstraintsEx" extensions="false" codeSystem="3.3.3.3.90"
      />
    </feature>
    <derives xref="InfoWorld.HL7.ITS.CR" />
  </target>
</feature>

```

```
</target>
</feature>
```

This example is used in the header for specialitateaMedic and CasaDeAsigurari for referral. An example of CDA modelled with the above SIMP:

```
<code codeSystemVersion="1" displayName="healthcare provider" codeSystemName="RoleClass"
      code="PROV" codeSystem="2.16.840.1.113883.5.110">
  <qualifier>
    <name code="394658006" codeSystem="2.16.840.1.113883.6.96" />
    <value displayName="Alta specialitate" codeSystemName="SpecialitatiMedicale"
          code="56" codeSystem="10.10.10.10.40">
      <qualifier>
        <value displayName="MEDIC PRIMAR" code="3" codeSystem="10.10.10.10.41" />
      </qualifier>
    </value>
  </qualifier>
</code>
<code codeSystemVersion="1" displayName="underwriter" codeSystemName="RoleClass"
      code="UNDWRT" codeSystem="2.16.840.1.113883.5.110" />
  <value displayName="COMP ASIG" codeSystemName="InsuranceHouse"
        code="CAS" codeSystem="3.3.3.3.90" />
</code>
```

On the code we have a qualifier which in turn has a qualifier on value (which only uses the value). Code still has a full qualifier. Differentiation between the two major qualifier done using the CD on name using the Vocabulary + DefaultCD constraint on that attribute. For the nested qualifier another association was added to the first association (`<feature xsi:type="AssociationEx" name="valoare" conformance="A" mandatory="false" min="0" max="1" rimName="value.qualifier">`) which has to be immediately after the connecting attribute (`<feature xsi:type="AttributeEx" name="valoareAtribut" conformance="A" mandatory="false" min="0" max="1" datatype="CD" rimName="value" mandatoryForAssociation="true">`)

5) Ex5 – Constrain constraintItem on 2 or more associations:

```
<feature xsi:type="AssociationEx" name="entryRelationshipCOMP" conformance="A" mandatory="true"
      min="1" max="1" rimName="entryRelationship">
  <target xsi:type="ConstrainedClassEx" name="EntryRelationshipCOMP" abstract="false">
    <feature xsi:type="AttributeEx" name="typeCode" conformance="A" mandatory="false" min="0"
          max="1"
          datatype="x_ActRelationshipEntryRelationship" editable="false" rimName="typeCode"
          default="COMP" defaultField=".">
    </feature>
    ...
    <derives xref="InfoWorld.HL7.CDA.EntryRelationship" />
  </target>
  <constraintItem path="typeCode" value="COMP" />
</feature>

<feature xsi:type="AssociationEx" name="entryRelationshipRSON" conformance="A" mandatory="false"
      min="0" max="1" rimName="entryRelationship">
  <target xsi:type="ConstrainedClassEx" name="EntryRelationshipRSON" abstract="false">
    <feature xsi:type="AttributeEx" name="typeCode" conformance="A" mandatory="false" min="0"
          max="1" datatype="x_ActRelationshipEntryRelationship" editable="false"
          rimName="typeCode"
          default="RSON" defaultField=".">
    </feature>
    ...
    <derives xref="InfoWorld.HL7.CDA.EntryRelationship" />
  </target>
  <constraintItem path="typeCode" value="RSON" />
</feature>
```

For the SIMP section above a constraint on the two entryRelationships from SubstanceAdministration is desired, one with typeCode = COMP and the other with typeCode = RSON. The need for this constraint appears when the translation CDA-SIMP is done. Because in these entryRelationships there are no CD attributes with Vocabulary and DefaultCD constraints, the translator will not be able to determine upon loading the CDA in which order and where these entryRelationships are to be placed, so the constraintItem is required. The only thing that's different is typeCode. So constraintItem is added on both associations. The the constraintItem path can contain any relative path from HL7CDA, relative to the association (in this example typeCode is on entryRelationship).

Generating .net classes for SIMP packages

The XML that describes the SIMP template at compilation of the package is translated into .net classes. A .net project is generated. This project is built (msbuild .net 3.5) and the resulting assembly represents the binary form of the SIMP package. Upon closing SharpDevelop this assembly is added to the package.

Translating this XML into .net classes is necessary so that the embedded RLUS metadatabase reasoner can correctly execute the semantic signifiers for each of the pilot scenarios.

The generation of .net classes is done upon the compilation (build) of SIMP package. At this point, the XML is parsed and all classes are extracted. It is very important that the direct editing of XML does not render it corrupted (invalid XML), as parsing will fail and so will the SIMP build. The classes are creating using "InfoWorld.HL7.Templates" + value in Template.Model.sourceModel namespace. The SIMP model is the main class in this namespace, which has the model's name (Template.Model.Name).

- Each Association (AssociationEx) will translate into a .net class. This class implements the INotifyPropertyChanged interface. PropertyChanged Event is thrown from each setter of the properties of the generated class. The connection between the two associations is done via C# properties

- Each attribute (AttributeEx) translates into a property (C#) on the parent class (AssociationEx)

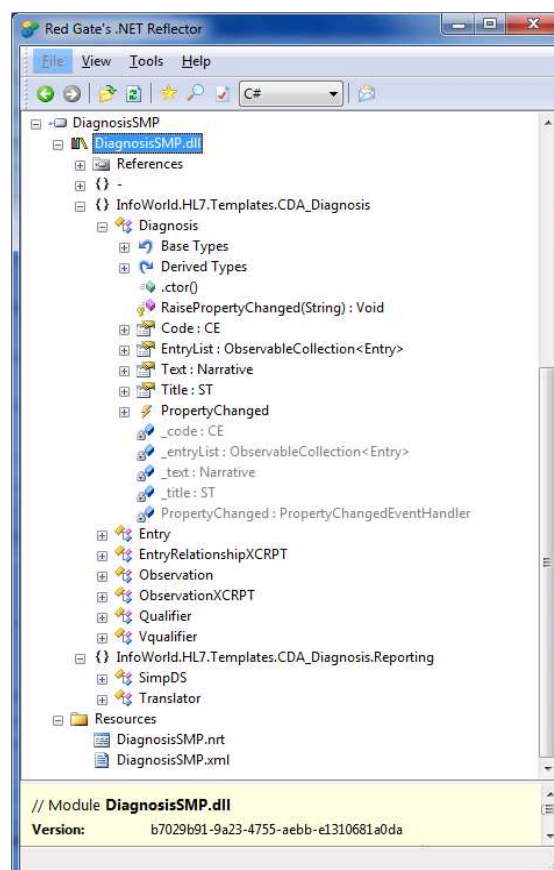


Fig. Example of classes generated for SIMP : DiagnosisSMP

Eg: For the diagnostics SIMP the generated classes are:

- The model is the main class , it implements an empty interface ISimp

```
[SectionID("2.16.840.1.113883.6.1", "11496-7"), XmlRoot(Namespace = "urn:hl7-org:v3",
IsNullable = true, ElementName = "Diagnosis"), XmlType(Namespace = "urn:hl7-org:v3",
TypeName = "Diagnosis"), GeneratedCode("aFine Code Generator", "1.0.0000")]
public class Diagnosis : INotifyPropertyChanged, ISimp
{
    ...
}
```

- The Title attribute on the model becomes a property in Diagnosis class

```
[DefaultAttribute("text", "Diagnostic"), Editable(false), IsMandatory,
XmlElement(ElementName="title", Order=1)]
public ST Title
{
    get
    {
        return this._title;
    }
    set
    {
        if (this._title != value)
        {
            if (value == null)
            {
                throw new ArgumentNullException("Title");
            }
            this._title = value;
            this.RaisePropertyChanged("Title");
        }
    }
}
```

When on a Diagnosis object's „Title“ property is updated through the setter, any handler attached to „event PropertyChangedEventHandler PropertyChanged“ will be invoked with the parameter „Title“.

The Entry association on the model is translated as a new class, connected to the Diagnosis class through a property. As the cardinality is greater than 1, the connection property is a specialized collection, in this case `ObservableCollection<Entry>`:

```
-
[XmlType(Namespace="urn:hl7-org:v3", TypeName="Entry"), XmlRoot(Namespace="urn:hl7-org:v3",
IsNullable=true, ElementName="Entry"),
GeneratedCode("aFine Code Generator", "1.0.0000")]
public class Entry : INotifyPropertyChanged
{
    // Fields
    private Observation _observation;
    private PropertyChangedEventHandler PropertyChanged;

    // Events
    public event PropertyChangedEventHandler PropertyChanged;

    // Methods
    public Entry();
    protected void RaisePropertyChanged(string propertyName);

    // Properties
    [XmlElement(ElementName="observation", Order=0)]
    public Observation Observation { get; set; }
}
```

In these class we have also the class Observation referenced with the property that has the same name because Observation is an association in the SIMP tree directly on Entry. In Diagnosis class the link is made using the EntryList property:

```
[XmlElement(ElementName="entry", Order=3)]
public ObservableCollection<Entry> EntryList
{
    get
    {
        return this._entryList;
    }
    set
    {
        if ((this._entryList != value) && (value != null))
        {
            this._entryList = value;
            this.RaisePropertyChanged("EntryList");
        }
    }
}
```

For cardinality greater than 1 an `ObservableCollection<T>` is used, as it can have attached handlers for the `PropertyChanged` and `CollectionChanged` events, and thus knowing when the collection or an item in the

collection was changed (through direct reference). For the collections, upon code generation, an element is automatically inserted into the collection. In the example above, this translates into the constructor of Diagnosis class:

```
public Diagnosis()
{
    ...
    this._entryList = new ObservableCollection<Entry>();
    try
    {
        this._entryList.Add(new Entry());
    }
    catch (Exception)
    {
    }
    ...
}
```

Attributes on the generated classes in SIMP

Attribute	Description
SectionIDAttribute	Attribute in the generated class Template.Model containing code and codeSystem for indentifying this section. Ex: [SectionID("2.16.840.1.113883.6.1", "11496-7")]
IsHeaderAttribute	Attribute that appears on the generated class from Template.Model when SIMP refers to CDA header and does not contain a certain section. Ex: [IsHeader]
SectionAttribute	Attribute in the generated class Template.Model containing the path for translating from SIMP to CDA. Ex: [Section("/ClinicalDocument/component/structuredBody/component[section/code/@code='48765-2' and section/code/@codeSystem='2.16.840.1.113883.6.1']/section")]
EditableAttribute	Attribute that appears on the properties C# that are generated from attributes. Equivalent of the attribute editable on an AttributeEx node from XML of SIMP
IsMandatoryAttribute	Attribute that appears on the properties C# that are generated from attributes. Equivalent of the attribute mandatory on an AttributeEx node from XML of SIMP
DefaultAttributeAttribute	Attribute that appears on the properties C# that are generated from attributes. Equivalent of the attribute default and defaultField .
CDValueAttribute	Attribute that appears on the properties C# that are generated from attributes type CD. Equivalent of node defaultCD on AttributeEx features
VocabularyAttribute	Attribute that appears on the properties C# that are generated from attributes type CD. Equivalent of node vocabulary on AttributeEx features

6. RLUS Wrapper for process mapper

The Process Mapper makes use of the ReMINE metadatabase to retrieve and store information of the patients in clinical documents. RLUS Wrapper WS provides the basic functionality to handle all the operations in the metadatabase: create, retrieve, update and delete as a wrapper of the RLUS service. It includes the common business logic executed on the RLUS client in the form of an encapsulated web service.

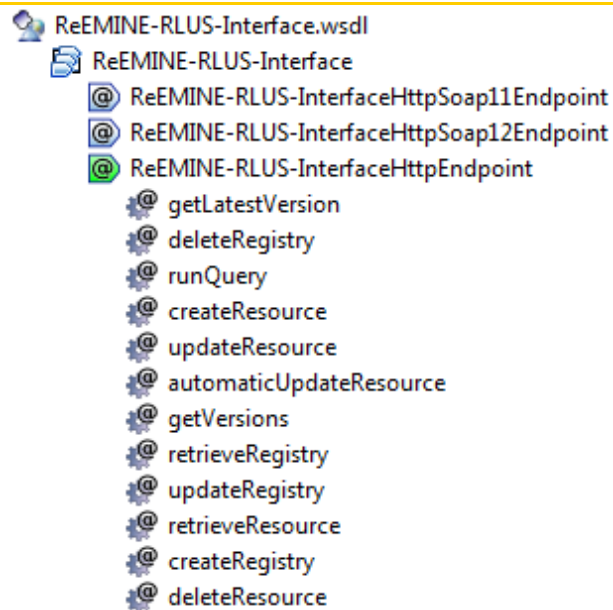


Figure 10 – RLUS Wrapper WSDL

We can divide the operations of the Process Mapper with ReMINE metadatabase in two main groups:

- Messages from HIS (like patient admission)
- Data storage

Messages from HIS

The messages coming from HIS, like when a patient arrives to the hospital, DAL/DEMS sends a message to the Process Mapper with the identification of the corresponding CDA. The process access to the metadatabase and retrieves all the needed information to keep on going.

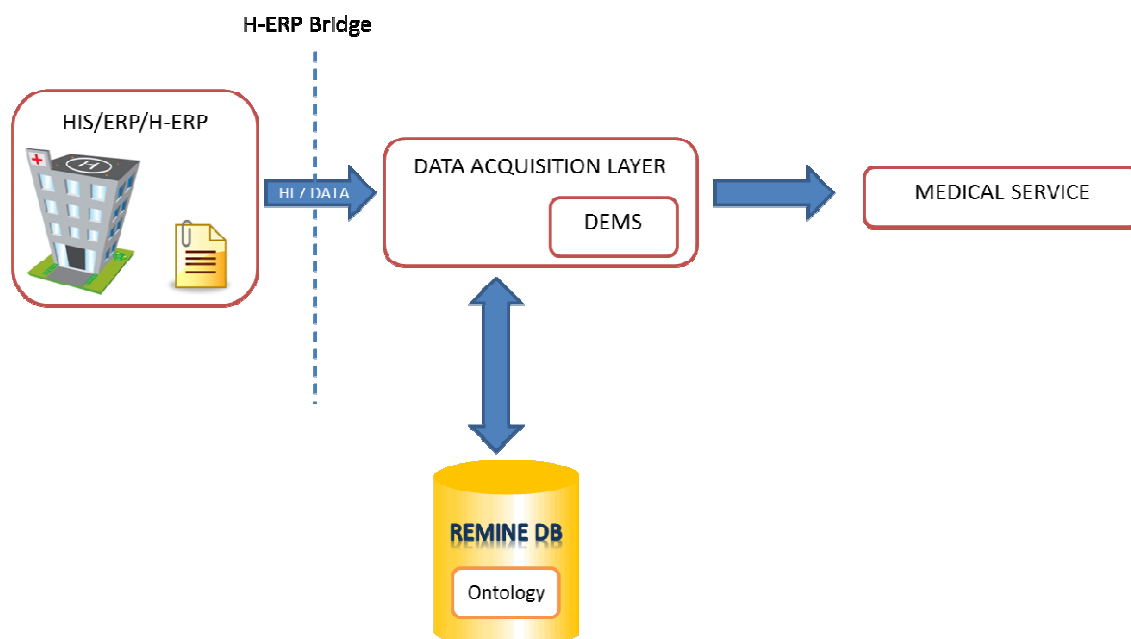


Figure 11 – DEMS stores the CDA and sends the identification to the medical service

Data storage

At particular points in the process flow, the Process Mapper stores all the relevant information in the corresponding CDA of a given patient. With this aim some WS to get the information and format them into standard CDA's has been developed. They also are in charge to call the RLUS-WS to store all the information, by calling the proper operations.

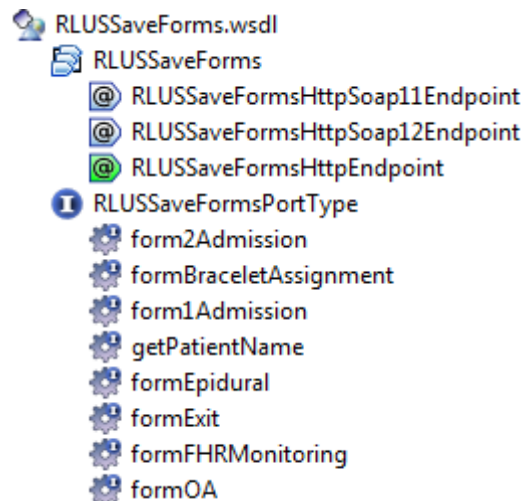


Figure 12 – Example of WS for forms storage (Sacco scenario)

Clinical Document	
Patient: [REDACTED]	MRN: ZZERTI26E57G131R
Birthdate: May 17, 1926	Sex: Female
Consultant:	Created On: August 26, 2010
Patient Admission	
Active Labour	
<ul style="list-style-type: none">• NulliParous• Absence of Contractions ≥ 2 in 10 minutes, regular and painful• Cervical length: 2 %• Dilatation: 2.0 cm• Result: Not active labour	

Figure 13 – Small example of CDA with style-sheet applied (xsl) (Sacco scenario)

7. Conclusion

Deliverable D.3.6 Third Revision of the WP3 framework represents the last steps that were taken for WP3 framework in the form of WP3 service enhancements:

- Audit service completion: all WP3 services are audit trailed on their main operations
- WS-Eventing development on RLUS service to be used in the data mining and knowledge inference module
- Completion of the metadatabase reasoner in the form of SIMP designer (designer for the semantic signifiers) and the reasoner itself as an executer for these templates inside the RLUS implementation
- RLUS Wrapper Web Service for the Process Mapper that encapsulates RLUS client business logic

8. Glossary

ADL - Archetype Definition Language

API – Application Programming Interface

CDA - Clinical Document Architecture

DLL – Dynamic Linked Library

GUI – Graphical user interface

HL7 - Health Level Seven

IDE – Integrated Development Environment

IHE - Integrating the Healthcare Enterprise

RIM - Reference Information Model

RLUS - Retrieve Locate Update Service

SNOMED CT - Systematized Nomenclature of Medicine-Clinical Terms

XML - Extensible Markup Language