ICT-2011-288048

EURECA

Enabling information re-Use by linking clinical REsearch and CAre

IP
Contract Nr: 288048

Deliverable: 7.2 Initial EURECA Security and Privacy Services

Due date of deliverable: 01-02-2013
Actual submission date: 14-02-2013

Start date of Project: 01 February 2012 Duration: 42 months

Responsible WP: Custodix

Revision: final

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

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Kristof De Schepper	Custodix	kristof.deschepper@custodix.com
Cédric Vansuyt	Custodix	cedric.vansuyt@custodix.com
Brecht Claerhout	Custodix	brecht.claerhout@custodix.com

0.2 Documents history

Document version #	Date	Change
V0.1	27/08/2012	Starting version, template
V0.2	27/08/2012	Definition of ToC
V0.3	10/01/2013	Initial draft
V0.4	15/01/2013	Revision
V0.5	25/01/2013	Revision
V0.6	05/02/2013	Final internal review
V1.0	14/02/2013	Approved Version to be submitted to EU

0.3 Document data

Keywords	security, privacy, services, eDPF, architecture
Editor Address data	Name: Cedric Vansuyt Partner: Custodix Address: Kortrijksesteenweg 216 b3 B-9830 Sint-Martens-Latem Phone: +32 (0)9 210 78 90 Fax: E-mail: cedric.vansuyt@custodix.com
Delivery date	14/02/2013

0.4 Distribution list

Date	Issue	E-mailer
14.02.2013	1.0	Benoit.ABELOOS@ec.europa.eu

Table of Contents

1 Introduction

The scope of the EURECA project is documented through different scenarios and use cases. Deliverable *D1.2: "Definition of relevant user scenarios",* based on input from users, contains a first iteration of the EURECA use cases and is a good illustration of the variety of issues addressed by the various applications of EURECA. The technical use cases describe the interaction between services and give a clear overview of the different data flows within EURECA and as such allow the overall security and privacy protection needs to be distilled from them.

Deliverable *D7.1 "Initial EURECA legal and ethical requirements"* contains a first (but already thorough) analysis of these security and privacy needs. This deliverable *D7.2: "Initial EURECA Security and Privacy Services"* focuses on the technical components required to address them.

The security services presented in this deliverable are a core part of the EURECA architecture described in deliverable *D2.2: "Initial EURECA architecture"*. Thus, all EURECA compliant services will be compatible with the presented security framework. More specifically this means that EURECA web services (for service-to-service communication) should be able to extract and interpret EURECA security credentials embedded in the web communication using the Web Services Simple Object Access Protocol (WS-SOAP). Security credentials are embedded according to widely accepted web standard, more particularly WS-Security. Services should also be capable to interact with a EURECA compliant authorisation service, i.e. compose an eXtensible Access Control Markup Language (XACML) access request, interpret it and respond to it. Different software modules will be delivered as part of the EURECA solution for easily integrating this functionality in web services.

This document is structured as follows. In the next section (Section 2) a short overview is given of the technical implications of the analyses of D7.1, and how the security architecture fits the overall EURECA architectural design principles. Section 3 gives an initial description of the security services to be developed in EURECA. In combination with section 4 which describes a number of specific data protection compliance oriented services, this forms the core of this deliverable. Finally, a summary is presented.

2 Scenario Security Requirements

2.1 Introduction

The EURECA privacy and security requirements have been analysed in deliverable *D7.1 "Initial EURECA legal and ethical requirements".* That deliverable shows that the broad application scope of the EURECA services imposes different data protection requirements on them, depending on their application domain. Furthermore data security and privacy are not only important during the operational phase (i.e. the exploitation phase or piloting phase; when the services are used in a real life environment) of EURECA. Also during the execution of the project itself, i.e. the research work (the building phase in D7.1), it is important that EURECA acts compliant to data protection regulations when working with real data coming from the clinical partners (e.g. for building clinical models, testing data mining algorithms or further designing the Common Information and Data Model CIM/CDM, etc.).

The following sections give a short overview of the security requirements of the different EURECA phases and application domains.

2.2 Build-up Phase: Anonymisation

A lot of the research within EURECA requires the availability of real medical data, for example the development and testing of data mining applications or the tools for semantic integration.

Legislation and regulation make the usage of sensitive personal data for these purposes very demanding. However, when data are properly de-identified (i.e. they can be considered anonymous as opposed to personal in legal understanding), this changes. Within EURECA a framework for compliant data sharing has been designed and extensively described in deliverable *D7.1 "Initial EURECA Legal and Ethical Requirements"*. The technical aspect of this framework encompasses tools for de-identification. For this, the project can mainly rely on work previously done on CAT and CATS (Custodix Anonymisation Tools and Services).

2.3 Exploitation Phase: Different Legal Domains

It was previously mentioned that EURECA targets a large range of different applications. A first analysis of the different application scenarios with respect to legal and security requirements has led to the identification of three main application domains in which different laws and regulations are determining for the security requirements:

1. Care domain
2. Research domain
3. Trial Support/Execution domain

Within the domain of care, the focus is on direct patient care. In this setting the treating physician collects information about the patients and uses this data in the course of medical treatment. The physician is in direct contact with the patients and has the patient's consent for treatment. This consent encompasses also use of the personal patient data for the treatment aims and management of healthcare. The main concern

in this domain is that the information is kept confidential. Data need to be personal (non-anonymous) and can only be accessed by authorised persons, who are furthermore under the confidentiality obligation. This domain will thus mainly rely on the "classical" authentication, authorisation and audit (AAA) functionality of the EURECA framework.

In the second domain, the research domain, interest shifts from the individual level to the cohort level. There is no need for a relation with the patient anymore, focus is on data analysis. The situation is the same as that of the data collection during the project build-up phase. One can rely on a similar techno-legal framework and on the same de-identification tools, be it that their usage is more dynamic (real-time de-identification of new datasets) and should be automated (e.g. de-identification built into a workflow).

Finally, EURECA addresses also scenarios related to clinical trials. These scenarios span finding patients for trials, enrolling them, managing their data throughout a trial, etc. These "trial support/execution domain" applications typically require a specific approach with respect to data protection. Next to the implementation of AAA security functionality, data flows often need to be enhanced with specific privacy protecting measures (cf. the protocol feasibility scenario[1]). Many of these scenarios are quite novel also from a legal perspective, considerations on those legal aspects can be found in the aforementioned D7.1. In addition, the ways of addressing those novel aspects will be further researched during the project lifetime. Technical solutions in these scenarios will be rather application specific.

2.3.1 Design Principles

The EURECA project addresses re-use of available data and the linkage of the care and clinical domain in a broad range of applications. The different types of applications require a flexible architecture, which allows the same tools to be used in different deployments (geo-spatial or logical) and different governance structures. As such EURECA aims to provide a framework of tools which can be easily interconnected in different configurations, tailored to the needs of different environments and end-users.

The EURECA architecture allows constructing solutions for the entire deployment spectrum. For example:

- A hospital may want to offer its physicians a convenient way to analyse their patient population or look for cases in the hospital database which (disease-wise) match their current patient.
- A medical consortium (e.g. the Breast International Group – BIG) or a pharma company might want to set up a large infrastructure (connecting multiple hospital sites) for conducting protocol feasibility studies.

Both scenarios rely on the same EURECA backend solutions, i.e. the semantic integration layer offering uniform access to EHR data and a clinical data query engine (in a sense one could even use the same cohort selection end-user application in both cases). However the former scenario requires deployment only in a single hospital, while the latter requires a cross-organisational deployment (and governance). One can

[1] In this scenario query services should only return aggregated query results. At the same time one might want to introduce specific privacy controls to ensure that consecutive aggregate results can also not lead to identification of individuals.

imagine that in the first scenario security requirements would for a part be covered by the confined environment (the hospital infrastructure) in which the tool is accessible. For the larger scale network, this would not be the case and a full featured security solution needs to be deployed (actually the particular use case also requires additional privacy measures[2]). The technical core required to implement these scenarios are the same; deployment topologies, data flows and governance are however different.

Technically, EURECA aims to obtain this flexibility through loose coupling and service orientation. Focus is thus on interoperability and interfacing. Modules built within the project should seamlessly operate through well specified interfaces on different levels (i.e. interoperability on the level of IT-protocol, data format, information content, etc.). With respect to security, this means that all EURECA services should easily integrate with the EURECA security framework, i.e. they will be capable of interpreting EURECA credentials coming from a EURECA compliant identity provider, or can be connected to a EURECA audit service.

The security framework itself follows the same design philosophy of componentising and loose coupling. This means that one is not required to deploy the complete security solution if one wants to deploy secured EURECA services. Authentication, authorisation and audit are functionalities provided through different services. These services can be independently used, but also interoperate (i.e. for example the authorisation service can interpret credentials provided by the authentication service).

Finally, the loose coupling combined with the usage of widely accepted international standards promotes the use of the EURECA components outside of the project's domain. On the other hand, it also avoids EURECA (internal) vendor lock-on, as effort to replace a specific security component is limited as much as possible through these principles.

2.3.2 Specific Requirements

2.3.2.1 Flexible Deployment

As explained EURECA aims to be a flexible framework (toolset) which allows to easily build solutions (for translational data usage and re-use) tailored to the need of a specific environment. The flexibility that is expected from the EURECA architecture also affects the security layer. The EURECA approach offers the flexibility to interconnect any independent EURECA compliant configuration, as the security layer is a core (required) aspect of the EURECA architecture, the security should not hamper this. What this exactly means with respect to required functionality is explained in section 3.7.3 after the overview of the security framework (for clarity).

2.3.2.2 Single Sign-On (SSO)

The EURECA platform provides multiple services, which require authentication of the user. A user will not want to go through the hassle of authenticating him/herself for every single EURECA service individually. Instead the user wants to access multiple EURECA services while authenticating him/herself only once. Single sign-on (SSO) offers a solution for this. SSO is a web-based session/user authentication process that

[2] i.e. no detailed data should leave hospital sites, thus only aggregated query results should be returned to a central protocol feasibility application.

permits a user to authenticate him/herself to multiple independent service providers by having to provide his/her credentials only once. The inverse process of SSO is called single logout (SLO). Section 3.3.1.1 elaborates on the SSO process.

2.3.2.3 Credential Forwarding and Delegation

Within the EURECA framework services might make use of other EURECA services (chaining of services). As these services will require authentication, user credentials of the requester need to be provided to all involved services. As mentioned before (see 2.3.2.2), the user experience can be enhanced by not having to re-authenticate him/herself, but rather to have intermediary services forward the requester's credentials to the requested service.

Forwarding identity assertions is a form of impersonation in which an assertion is directly reused by an intermediary service to impersonate a subject from whom an assertion was obtained. Delegation moves beyond the forwarding scenario by adding information to the assertion that explicitly identifies all the parties through which a transaction flows. Both techniques are described in 3.3.1.2.

3 Main Security Architecture

3.1 Overview / Main Components

The EURECA security framework will consist of modular re-usable components dealing with authentication, authorisation, auditing and de-identification. In this section, the overall framework providing authentication, authorisation and audit functionality is discussed. Figure 1 illustrates how the different components logically integrate with each other.

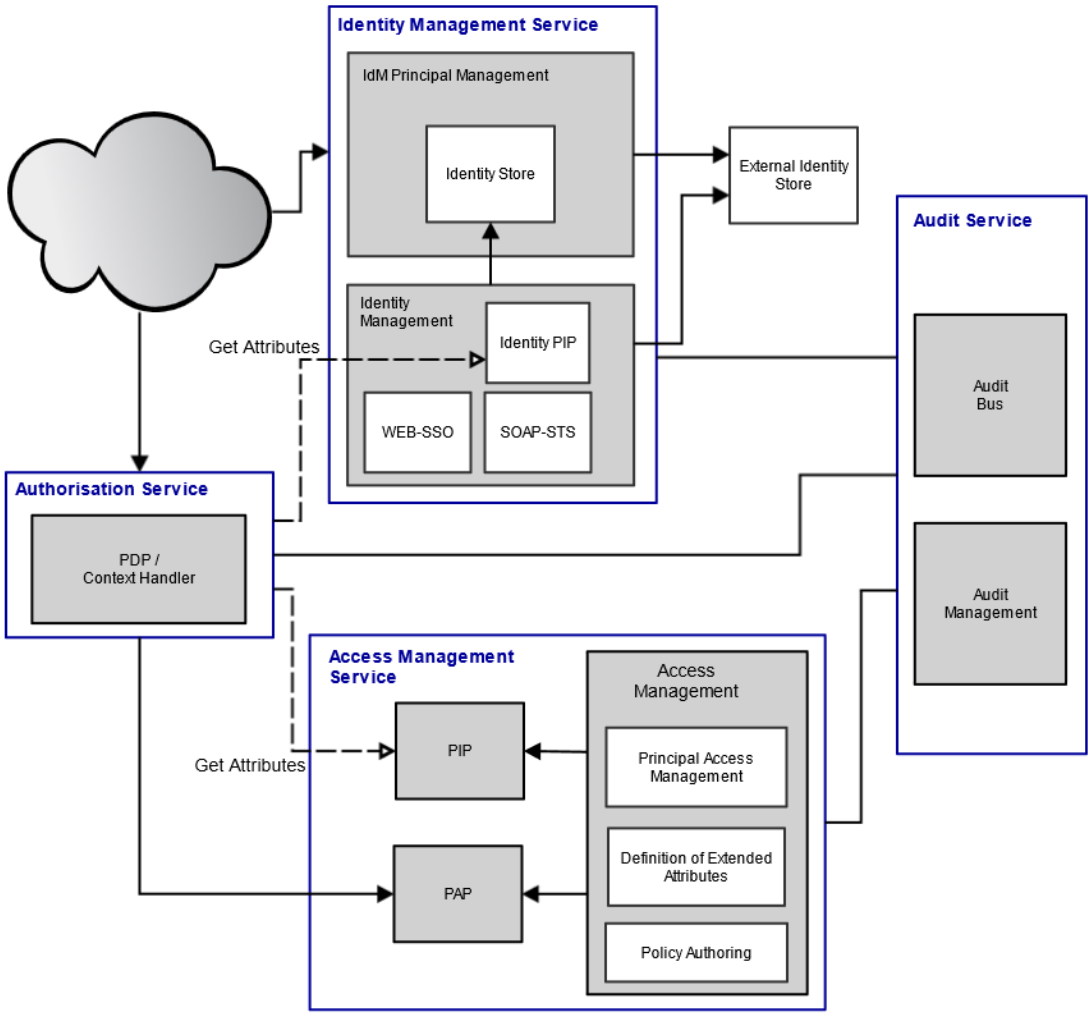

Figure 1: Initial EURECA security framework

As previously mentioned, the actual deployment is irrelevant for this explanation. The security components can be used to easily provide security functionality on a local install, or to ensure confidentiality on a large trans-national distributed network. At the same time, not necessarily all components need to be instantiated in order to benefit from the EURECA security framework. One might for example opt to omit the authorisation server from a local implementation and leave authorisation to the internals of the application, while still relying on the user management provided by the Identity Manager.

Figure 1 shows the main components of the EURECA security framework:

1) **Identity Management Service**
   The identity management service combines the functionality of principal management and identity provision. Principal management involves the management of the digital identity of the main actors in a secured environment. These principals mainly consist of individual users (humans) and services ("computers"). The identity provision component deals with assigning identities which can be understood by all services on the secured environment to principals so they can authenticate themselves to the Identity Provider (IdP).
2) **Access Management Service**
   This component deals with the authoring and management of different access control policies that need to be enforced across the security infrastructure.
3) **Authorisation Service**
   This is a support component which evaluates access requests issued by applications according to the access control policies and requestor security context (i.e. principal credentials, etc.).
4) **Audit Service**
   A concentrator for security log information.

The following sections give an initial description of the majority of these components.

3.2 Principal Management

The principal management will keep track of the users and services in EURECA by storing them in a hierarchical tree of domains and organisations.
Domains are used to group principals. A domain is globally identified by its globally unique identifier (GUID). Domains are mutual exclusive. This means a principal can only be part of one domain. Since a domain is uniquely identified by its GUID, this GUID is used as reference by principals and organisations. The IdM environment contains at least one domain, and each domain can contain zero, one or more principals.

While principals are already grouped by domain, it is often useful to group a subset of principals that belong to the same domain. This subset can be grouped in an organisation. An organisation is member of one domain, which is referenced by the domain GUID field.

To manage all of the EURECA principals, LDAP (Lightweight Directory Access Protocol) is used. LDAP is an application protocol that can be used to access and maintain distributed directory information services over an internet protocol network. The core of the protocol is defined by the Internet Engineering Task Force (IETF) in RFC4510[3]. The directory information services contain information that is organized in a hierarchical directory structure. This information can be queried and filtered so that only the required information is returned. LDAP directory is a "write once, read many times" service.

[3] IETF, LDAP Technical Specification Road Map, 2006, http://tools.ietf.org/html/rfc4510

LDAP is a good solution to use as user credential store as part of the user management services in EURECA (mainly due to the presence of the flexible password policies, which take a high implementation effort when building from scratch). For this the EURECA platform will include an already existing implementation of the LDAP protocol like OpenDS[4] or OpenLDAP[5].

3.3 Identity Management Service

The Identity Management (IdM) service is responsible for principal management and identity provision within EURECA.

3.3.1 Identity Provision

For the exchange of identity information an attribute-based approach is chosen, which means that user attributes (e.g., the user's name, his date of birth, his email address, his clearance level, the authentication assurance level, etc.) are exchanged in a standardised format called a security token. A security token is protected against forgery and replay attacks by adding a timestamp and having the issuer sign it.

Attribute-based is sometimes also referred to as claims-based as a signed token can be viewed as a claim by the issuer that the contained attributes have certain values. Attribute-based identity management (authentication, attribute release, authorisation) especially plays well in SOA environments like EURECA. Within a SOA environment application functionality is captured in services which are called by clients to achieve their goals. Since SOA demands the use of accepted, open standards (usually web-services based) to achieve interoperability between components it is relatively easy to bind attribute-based security token exchange.

A commonly used implementation method for identity provision including SSO (discussed in 2.3.2.2) and a security token specification is the Security Assertion Markup Language (SAML)[6] protocol. SAML is an XML framework for exchanging security information over the Internet. SAML enables disparate security services systems to exchange authentication and authorisation information between one or more security domains. Next to the single sign-on profile, SAML offers a rich set of other useful profiles relevant for authentication and authorisation.

For the issuing of security tokens the EURECA security platform will provide two issue components, an Identity Provider (IdP) and a Security Token Service (STS), each implementing a specific part of the EURECA required identity provision functionality.

The IdP component provides an implementation of the SSO browser base SAML profile[7]. If a user tries to access a protected resource of a service belonging to the EURECA platform through his **browser** (over HTTP(S)), he is redirected to the IdP

[4] OpenDS, Open Source Java LDAP Directory Service, http://www.opends.org/
[5] OpenLDAP, http://www.openldap.org/
[6] SAML 2.0, 2008, "Security Assertion Markup Language (SAML) V2.0 Technical Overview", version 2.0, available from: https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf. [1 February 2013]
[7] SAML profiles, 2005, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", available from: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf. [1 February 2013]

who will issue a security token after the user has authenticated him/herself to the system. This security token will then be validated by the service.

The STS component provides an implementation of the WS-Trust[8] specification. In this case a user wants to access a protected resource of a service belonging to the EURECA platform through an **application** (over SOAP). The STS will issue a security token after the user has authenticated him/herself to the system. This security token will then be validated by the service.

3.3.1.1 Single Sign-On

As mentioned in 2.3.2.2, Single Sign-On is required in the EURECA platform. SSO will be implemented in EURECA using the web browser SSO profile[9] of the Security Assertion Markup Language (SAML)[10]. As explained above, SAML (see deliverable *D2.1: "State of the Art Report on Standards")* is an OASIS XML-based standard for exchanging authentication and authorisation data between security domains. It uses security tokens containing assertions to pass information about a principle between a SAML authority, called identity provider, and a web service, called a service provider.

The SSO process is shown in Figure 2. A user (Requestor) wants to access a resource of a service provider (SP), through his browser, where he has no local active authenticated session. A HTTP request is sent to the service provider (step 1). Because the service provider does not have an authenticated session for the user, it issues an authentication request message to be delivered to the identity provider (IdP). This message is sent from the SP over the user's browser to the IdP (HTTP Redirect or HTTP POST) (see step 2).

[8] WS-Trust, 2007, "WS-Trust 1.3", available from: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf [1 February 2013]
[9] SAML profiles, 2005, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", available from: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf. [1 February 2013]
[10] SAML 2.0, 2008, "Security Assertion Markup Language (SAML) V2.0 Technical Overview", version 2.0, available from: https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf. [1 February 2013]

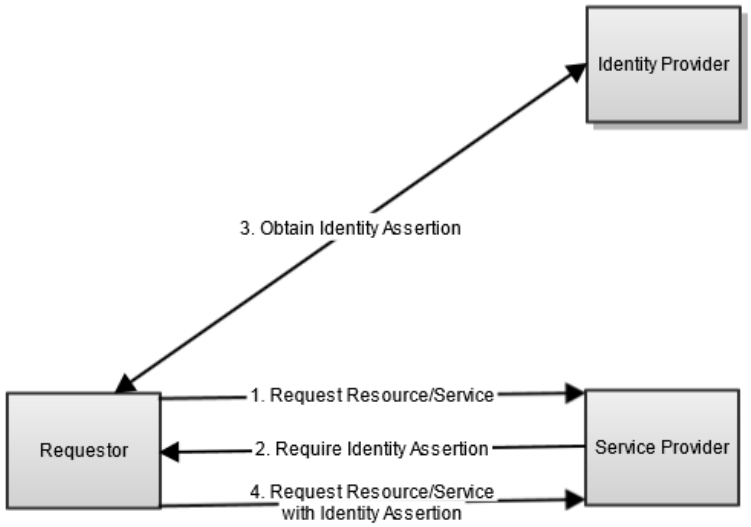

Figure 2: Single Sign-On Process

If the user has a running authenticated session on the IdP, it will provide an identity assertion for the user. This assertion is sent back in a response message to the SP (HTTP Post) (step 3). If no authenticated session was found, the user first has to authenticate him/herself by providing his/her credentials. Finally (step 4) the SP will verify the assertion of the IdP and grant the user access if it was valid (by creating an authenticated session for the user).

3.3.1.2 Credential Forwarding and Delegation

Through credential forwarding (Figure 3) a service A can access a service B on behalf of an end user U. As credential forwarding means that the identity assertion of U received by A is just forwarded as it is to B, the service B will think that the user U directly contacted him/her without knowing that there is a service A lying in between. Forwarding is not possible though if the identity assertion is limited in audience to A or encrypted for A. In the latter case service B would not be able to decrypt and interpret the assertion.

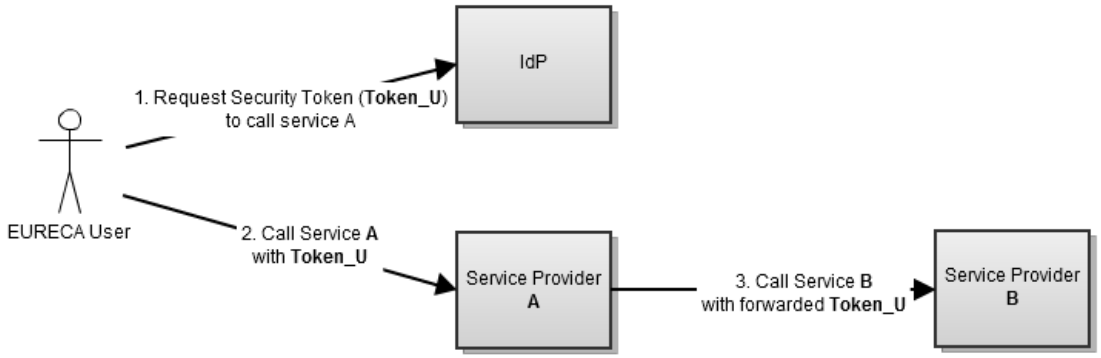

Figure 3: Credential forwarding

Delegation moves beyond the forwarding scenario by adding information to the assertion that explicitly identifies all the parties through which a transaction flows. While it is ultimately a matter for the identity provider as to how and on what basis this information is collected, it is often assured by routing requests back through the identity provider at each hop to cryptographically guarantee that each party has been authenticated and the appropriate policy is enforced. Figure 4 shows how the forwarding model is extended by adding a request back to the IdP by service A. While forwarding implies that the user identity token is forwarded by service A to service B, in delegation service A requests a delegation assertion from the IdP. This delegation assertion will then state the identity of the current client (service A) as a delegate, and the identity of the user on whose behalf the client is acting (end user U).

EURECA will use the SAML 2.0 delegation specification[11] to implement this functionality.

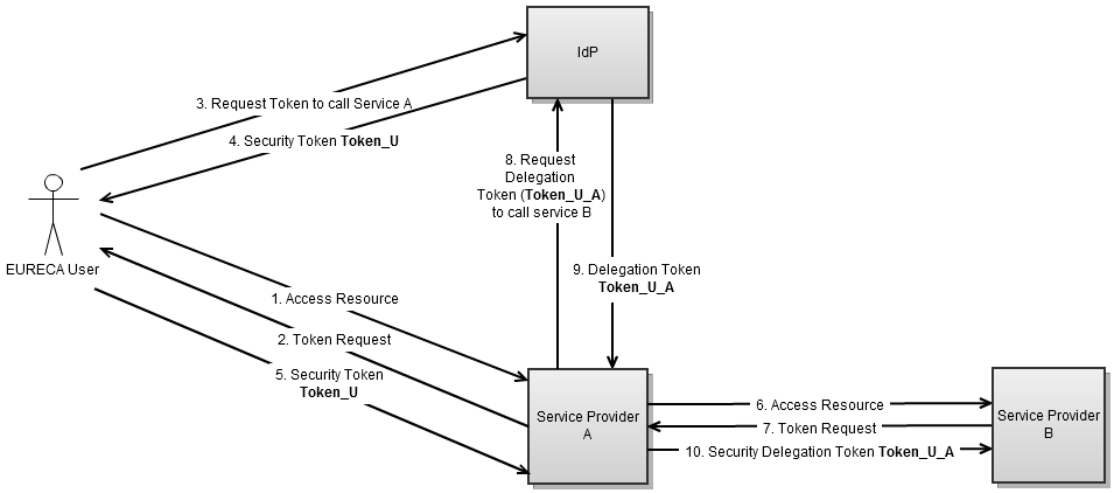

Figure 4: Credential delegation

3.4 Access Management Service

The Access Management Service consists of three main components, each providing specific access management functionality, that will be explained in the following subsections.

3.4.1 Policy Administration Point (PAP)

The Policy Administration Point (PAP) component is responsible for authoring and management of the different access control policies that need to be enforced in the EURECA framework. These access control policies are a formal set of rules that define what action, if any, a user can take on a particular resource. For example, reading (action) the medical file of patient A (resource) is restricted only to the threating physician B (user). Using these policies the authorisation service can make a

[11] SAML 2.0 delegation specification, 2009, "SAML V2.0 Condition for Delegation Restriction Version 1.0", available from: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cs-01.pdf. [1 February 2013]

decision (grant or deny access) on an incoming access request (described in section 3.5).

In large frameworks like EURECA, this authoring and management of policies can become very complex. That is why it is important to choose a suitable policy standard that supports the access requirements defined in the framework (whether with extensions or not). Several solutions are available for defining and enforcing complex policies, such as for example XACML[12], PERMIS[13], PONDER[14], Cassandra[15]. After a thorough evaluation of EURECA's access control requirements, it was decided to use XACML as the authorisation solution. XACML is primarily an Attribute-Based Access Control (ABAC) system. ABAC presents an access control model inherently capable of meeting many of the "modern" access control demands (e.g. data dependent access policies, environment dependent policies, etc.). In ABAC, attributes that are associated with a user, action or resource serve as an input to the decision whether a given user may access a given resource in a particular way.

The eXtensible Access Control Markup Language (XACML) is a XML based declarative access control policy language defining both a policy, decision request and decision response language. XACML contains several profiles for supporting Role-Based Access Control (RBAC), multiple resources, hierarchical resource, etc. The policy language is designed to offer great interoperability with different platforms and extensibility.

The smallest element in the XACML policy language is the rule element. Each rule targets a set of decision requests to which it is intended to apply. The rule target is expressed in form of a logical expression on attributes of the request. A rule also has a condition element that refines even more the set of decision requests that are targeted. If for the given target and condition a set of decision requests is found, the rule responses with a defined effect value that permits or denies access to the selected set of decision requests. Rules cannot exist on their own; they have to be combined in a policy element. This policy element groups rules and combines the access decisions (permit or deny), which were evaluated for the targeted set of decision requests, to a general policy access decision using a combining algorithm. A policy element can target a set of decision requests. An example of a policy element is shown in Figure 5.

Finally, there is an optional element called policyset. This element can group policy elements and/or other policysets. Like the policy element it has a combining algorithm to generate a general policyset access decision. The policyset can target a set of decision requests.

[12] OASIS, 2005, 'XACML: eXtensible Access Control Markup Language', Version 2.0, Available from: <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>. [16 July 2012]

[13] Chadwick, D, Zhao, G, Otenko, S, Laborde, R, Su, L, Anh Nguyen, T, 2008, 'PERMIS: a modular authorization infrastructure', Concurrency and Computation: Practice and Experience, vol. 20, no. 11, pp. 1341-1357

[14] Damianou, N, Dulay, N, Lupu, E, Sloman, M, 2001, 'The Ponder Policy Specification Language', POLICY 2001 Proceedings of the International Workshop on Policies for Distributed Systems and Networks, pp. 18-38

[15] Moritz, Y, Becker, PS, 2004, 'Cassandra: Distributed Access Control Policies with Tunable Expressiveness, Policies for Distributed Systems and Networks', POLICY 2004 Proceedings of the Fifth IEEE International Workshop, pp. 159 – 168

```
  <Policy PolicyId="SamplePolicy"
          RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining algorithm:permit-
overrides">
    <Target>
      <Resources>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">patientDiaryA</AttributeValue>
          <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
      </Resources>
    </Target>
    <Rule RuleId="AccessRule" Effect="Permit">
      <Target>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
            <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                       AttributeId="urn:actionid"/>
          </ActionMatch>
        </Actions>
      </Target>
        <Condition>
           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
              <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
                   <ResourceAttributeDesignator
                   AttributeId="urn:custodix:healthcare:0.1:patient-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" />
               </Apply>
                   <SubjectAttributeDesignator
AttributeId="urn:custodix:healthcare:0.1:attribute:hcp-relation:gmd-holder-of"
DataType="http://www.w3.org/2001/XMLSchema#string" />
            </Apply>
          </Condition>
    </Rule>
    <Rule RuleId="FinalRule" Effect="Deny"/>
  </Policy>
```

Figure 5: Example of a XACML policy element

3.4.2 Policy Information Point (PIP)

The Policy Information Point (PIP) is responsible for the resolution of possible missing attributes in an access control request coming from the Policy Enforcement Point (PEP). These attributes are needed by the authorisation service, i.e. the Policy Decision Point (PDP), to come to an accurate decision about the request. If this authorisation service needs further information to evaluate the incoming request from the PEP, the PIP is called, either directly by the authorisation service itself or by returning missing attributes to the PEP which have to invoke the PIP.

3.4.3 Access Management

The access management provides a user interface to the access control administrators that enable them to manage and author policies and attribute information in a user-friendly, intuitive way. Advanced user interfaces will be required in order to display the complex structure of the policies.

3.5 Authorisation Service

The authorisation service is responsible of evaluating access requests coming from the different applications that restrict their resources by access control. The authorisation service evaluates these requests based on the (AC) policies that were defined in the authorisation management service and the requester security context. As described in the previous section, it has been decided that the policies defined in EURECA will be XACML based. This means that by using these XACML based policies, the authorisation service should be able to validate XACML decision request messages sent by Policy Enforcement Points (PEPs). The component responsible for the validation (making access decisions) is called the Policy Decision Point (PDP).

Next to a PDP, a context handler component is part of the authorisation service. This component will augment the decision request messages coming from the PEPs with additional (probably missing) attributes obtained from Attribute Authorities (AAs) or Attribute Repositories. The AAs are accessed through a Policy Information Point (PIP) (see above).

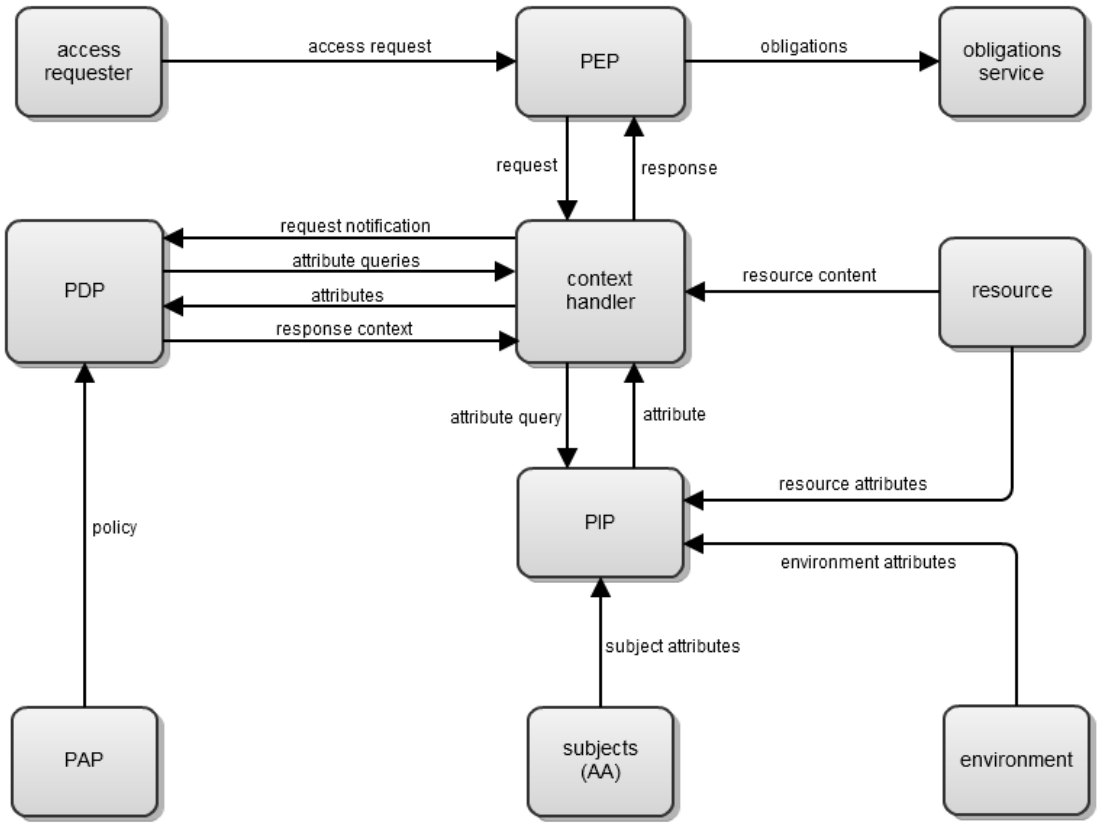

Figure 6: XACML data flow diagram

The general flow is straightforward. The PEP intercepts access requests by a principal on one or more protected resources on a server. The PEP generates a decision request based on the attributes of the subject, the resource in question, the performed action and other information pertaining to this access request. This decision request is send to the Authorisation Service. Here, the request is possibly augmented by the context handler component which passes it to the PDP. The PDP will interpret the request and searches for policies (coming from the PAP) that apply to the request. Based on the rules defined in the found policies, the PDP will make an access decision

and includes this decision in a decision response message. This message is sent back to the PEP. The PEP will use this response to decide if access is granted to the protected resource or not.

3.6 Audit Service

Every security framework contains a (centralised) audit service. Each authentication attempt (both successful and failed), resource access/change, available issue (e.g. server exceptions), etc. needs to be logged by the audit service. Next to the logging functionality, the audit service needs to present the logs to the administrators in such a way that they can be easily consulted and interpreted.

3.7 Security Services Integration

3.7.1 Security Integration

As mentioned before (see 2.3.1), the security framework is designed as a set of independently useable components with clearly specified interfaces. As such, service providers needing to make their service EURECA compliant (and thus also compliant with the security specifications) can integrate with the security framework by directly addressing the security services through their service interface (see Figure 7).

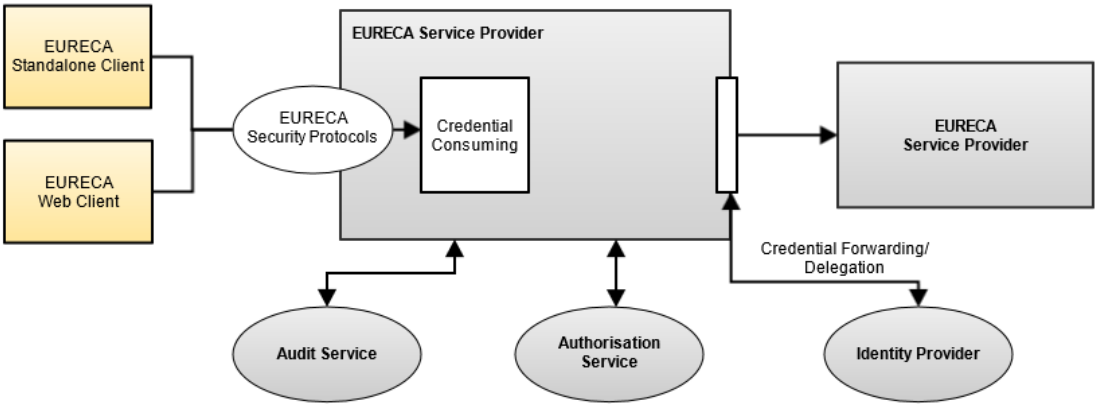

Figure 7: Integration of service provider

This is still a challenge as this requires implementers to deal with (non-exhaustive):

- secure communication (security protocols)
  - e.g. extracting security information from security protocols
- the service site logic required for complex functionality such as delegation
- interaction with access and audit services
  - e.g. creating valid access requests

However, because the EURECA security architecture aims to rely as much as possible on existing industry standards, a big part of this functionality, mainly the security protocol handling, can be easily implemented through standard or (freely available) third party libraries.

Development for the Microsoft ".NET" environment can rely on the Microsoft provided Windows Communication Foundation[16] (WCF) and Windows Identity Foundation[17]

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

(WIF) frameworks. The former is a bundled set of APIs for implementing distributed applications. It includes an implementation of many of the WS-* standards (including WS-Security).
The latter WIF framework aims at building identity-aware applications and abstracts the WS-Trust and WS-Federation protocols. It also supports the SAML 1.1 and 2.0 standards for generating authentication tokens.

Java developers can rely on a number of freely available libraries from different sources, such as Metro[18], CXF[19], OpenSAML[20], Sun XACML derivatives, etc. Metro is the Oracle web service stack which implements many of the WS-* standards, like WS-Security and WS-Trust. CXF[21] is an open source web service framework similar to Metro which integrates well with the widespread Spring Framework. Sun XACML was once the de-facto reference implementation of the OASIS XACML standard (supporting version 2.0). Although the project was abandoned, it served as a source for several spin-offs that are freely available, such as the JBoss XACML implementation.

Furthermore, in order to further facilitate creating EURECA compliant services, libraries and components that implement a large part of the security functionality will be provided for different platforms.

An example of how EURECA security modules will integrate in applications is shown on Figure 8. This figure shows a java web application relying on the Spring Framework. For ensuring authenticated communication and authorisation, the EURECA security library provides dedicated filters compatible with the Spring Framework. The authentication filter is basically a SAML consumer which extracts security information from authentication headers and saves it into the Spring security context. The authorisation filter will evaluate every incoming request by creating an access control request (compliant to the EURECA rules) and have it evaluated by an authorisation service. For creating this request, the authorisation filter relies on information gathered in the Spring security context.

[18] Metro web service stack: http://metro.java.net/
[19] Apache CXF: http://cxf.apache.org/
[20] OpenSAML: https://wiki.shibboleth.net/confluence/display/OpenSAML/Home
[21] Apache CXF: http://cxf.apache.org/

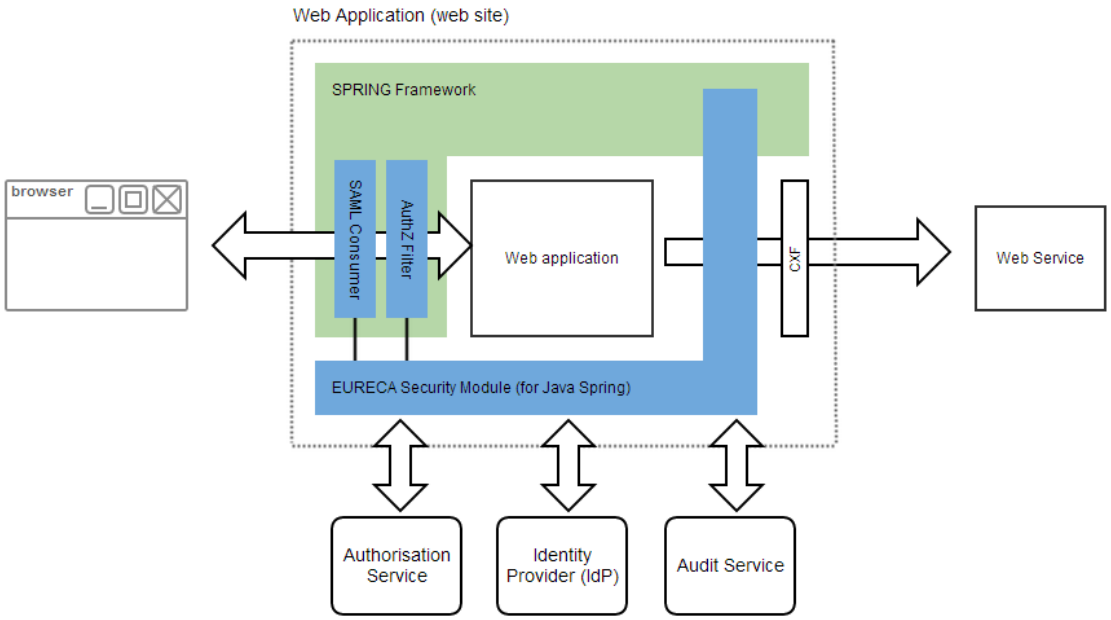

Figure 8: Example of integration of security modules

If this application needs to call other service providers on behalf of the user, the complex process of delegation (see 3.3.1.2), i.e. getting new credentials from an identity Provider, is dealt with by the EURECA security module (which on its turn relies on CXF for communication). The security module also takes care of logging towards a EURECA compatible log service.

3.7.2 Security Gateway

The EURECA security functionality can be bundled in a (reverse) proxy service. All security functionality will be provided by the security gateway that proxies connections to unsecure EURECA SPs. This allows EURECA SPs to be developed (more or less) without taking into account the EURECA security environment. As long as the proxy is the only path available to reach the SP from the internet, using a security proxy guarantees that there is no chance of bypassing the security controls. Towards service requestors, the new SP acts as a fully EURECA compliant SP.

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

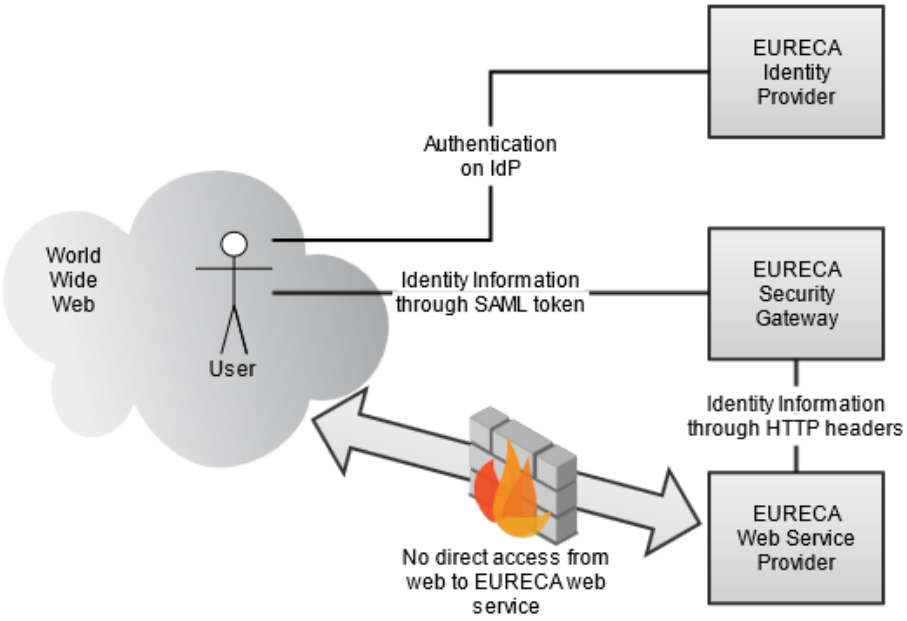

Figure 9: EURECA secure gateway proxy

The gateway will handle all SAML protocols for the EURECA service provider. A user passes his identity information to the security gateway through a SAML token. When there is no active authenticated session, the gateway redirects the user to the IdP for authentication.
The SAML identity token provided by the IdP is then interpreted and verified by the proxy. The proxy extracts the identity attributes from the SAML token, and adds them to the Authorization header that will be inserted in each HTTP request forwarded to the service provider. The SP can then extract these attributes from the header and use them to identify the end user without having to worry about the SAML protocols.

3.7.3 Security Framework Component Deployment

The objective of EURECA is to create a flexible framework (toolset) which can be deployed in different topologies to offer a solution for particular business cases in different domains (in the area of data re-use and translational science). The technical approach to reach this objective has been shortly revisited in section 2.3.2.1 together with the fact that this componentisation strategy is also valid for the security framework. The security framework will bring a solution to those building large infrastructures, needing to impose a unified security policy over a cross-organisation setup. At the same time it should be able to simplify the life of developers building a small scale solution (e.g. locally in a hospital).

With this flexibility as an objective of the EURECA security framework, the additional requirement arises that it should be possible to interconnect different built up security domains (small, large, etc.). This means that the security infrastructure should allow federation of different security domains. The security framework therefore needs to support a.o.:

- **Chaining of authorisation services**:
  This means that authorisation services should be able to forward requests to other authorisation services. In this way, access requests can be evaluated according to policies managed in different environments.
- **'Security vocabulary' mapping services (for security attributes):**
  Bridging (federating) different security domains means that security services in one domain should be able to 'understand' access requests from another domain[22]. In an attribute oriented security environment this implies mapping of various security attributes.
- **Identity federation**:
  Identity federation is in a sense a security vocabulary mapping of identities.

An example of the envisioned flexibility is given in Figure 10. On the left, a possible local install is shown. An application ('application A') relies on the EURECA semantic layer for data access. The services rely on a EURECA compliant authorisation service ('AuthZ Service A') and identity management service ('Identity Mgt Service A') for security. The complete install is locally deployed and managed.

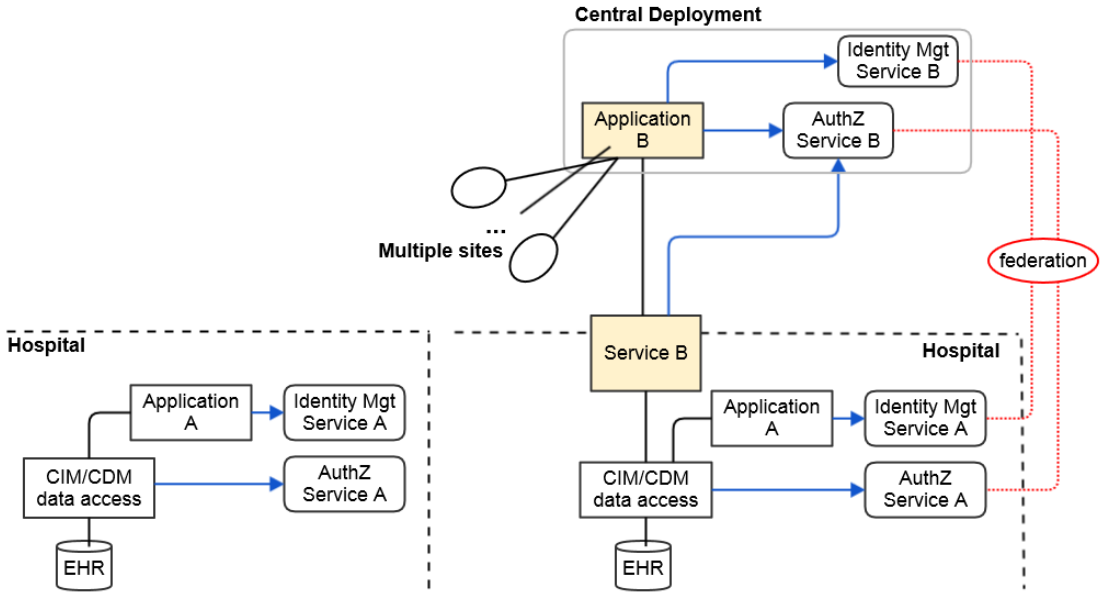

Figure 10: Security domain federation

On the right however, we see the same hospital which at some point in time joins an international collaboration for data sharing. This infrastructure is EURECA compliant, which means that hooking up to it should be as simple as installing 'service B' and connecting it to the data access layer. Logically the international collaboration has its own governance structure and thus its services connect to central authorisation services which enforce those cross-organisational policies. At the same time, the local hospital itself wants to ensure that proper access control is enforced and decides to

[22] With respect to attribute mapping, a domain doesn't necessarily need to be a governance domain. Attribute mapping is also used for translating attributes between the governance level (where policies are managed) and the implementation level (i.e. how a particular application internally deals with security attributes).

keep this at the level of the semantic layer. This means that for the data flow of the international collaboration, access control is governed both locally as centrally (by other organisations).

In order to obtain the level of flexibility illustrated, the security framework must be capable of federating different governance domains. In the example given, when someone uses ‘application B’ on the central platform, this will result in access requests at the local data access layer which are forwarded to ‘AuthZ Service A’. However, without specific measures for federation, that authorisation service will not be able to properly evaluate access requests which originated from users outside its security domain[23].

[23] In this particular example, one could configure the ‘AuthZ Service A’, to accept all requests originating from ‘Service B’. However this would mean that real local control over the access requests Is lost (all is delegated to the central governance structure).

4 Other Security Services

4.1 De-identification and Pseudonymisation

4.1.1 Data import Requirements

Sometimes patient information needs to be anonymised, when imported in EURECA. As anonymisation, i.e. by using k-anonymity, removes or generalises the information making it not useful for the project, the patient data will not be fully anonymised but rather be de-facto anonymised (further detailed information on de-facto anonymisation can be found in D7.1).

4.1.2 CATS

CAT (Custodix Anonymisation Tool) and its service-oriented evolution **CATS** (Custodix Anonymisation Tool Services) are responsible for the transformation of input files (plain text, CSV, XML ...) to output files. Based on a predefined set of transformation rules, called a privacy profile, CATS will process an input file and deliver it to the next component in chain (e.g. a database on a research platform). CATS supports multiple privacy profiles. Before processing a file, it will select the correct privacy profile based on the detected file type, and given content.

Important transformations are:

- Pseudonymisation: Based on person identifying information a pseudonym is added.
- De-identification: Person identifying information is cleared from the input.
- Encryption: Sensitive data can be encrypted with configurable public key.
- String replacement: Based on regular expression, string values can be replaced.

CATS is recommended as the de-identification tool during the project development phase as part of the privacy framework (see deliverable 7.1; section 6.2). All personal data coming from the data exporter will be first de-identified by CATS before it is stored in the EURECA datawarehouse(s) (see Figure 11). During this procedure each hospital-awarded pseudonym will be replaced by a random number (an anonymous identifier) without storing a link between both identifiers.

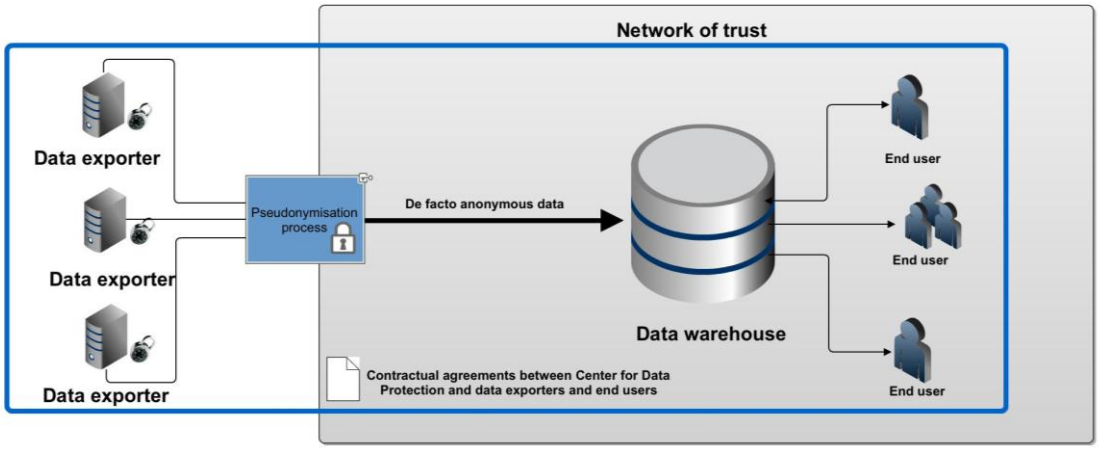

Figure 11: Development phase pseudonymisation

4.2 Patient Consent

Consent is an imperative precondition for the use of patient data within the EURECA framework. As a rule, whenever processing of patient data is taking place within EURECA, he/she must have agreed to it by giving consent prior to the processing (this is extensively described in D7.1). Before obtaining the patient´s consent, the patient was provided with all the information necessary to understand what will be done with his/her data. The requirement of informed consent will ensure that the use of their data is transparent to the patients and will not only prevent legal uncertainty, but also generate trust among the patients. Consent has been extensively investigated in the FP7 CONTRACT[24] project.

The absence or presence of consent for a specific matter translates into an authorisation decision. Patient empowerment can be enforced by introducing consent policies, configured by the patient himself. The consent the patient gives is automatically evaluated during authorisation by the PDP. The integration of consent into the authorisation framework can lead to improved assurance of compliance to consent directives (and thus to law) because of the automated enforcing.
The link between consent and authorisation will be researched in more detail later in the project.

[24] http://www.contract-fp7.eu/

5 Summary

This document describes the initial iteration of the EURECA security and privacy services. An overview was given of the main security requirements that have been analysed in deliverable D7.1 “Initial EURECA legal and ethical requirements”. Next an overview was given of the main security architecture that will be developed in EURECA. Finally a short overview was given of the de-identification and pseudonymisation services that will be used in EURECA.

6 Glossary

Abbreviation	Description
AA	Attribute Authority
AAA	Authentication, Authorisation and Audit
ABAC	Attribute -Based Access Control
AC	Access Control
CAT(S)	Custodix Anonymisation Tools (and Services)
CDM	Common Data Model
CIM	Common Information Model
EHR	Electronic Health Record
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
IdM	Identity Management
IdP	Identity Provider
IETF	Internet Engineering Task Force
LDAP	Lightweight Directory Access Protocol
OASIS	Organization for the Advancement of Structured Information Standards
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PERMIS	PrivilEge and Role Management Infrastructure Standards
PIP	Policy Information Point
RBAC	Role-Based Access Control
SAML	Security Assertion Markup Language
SLO	Single Log-Out
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SP	Service Provider
SSO	Single Sign-On
STS	Security Token Service
WS	Web Service
XACML	eXtensible Access Control Markup Language