Integrating the Healthcare Enterprise

IHE IT-Infrastructure White Paper

Access Control

Public Comment

Date: September 28, 2009

Authors: Jörg Caumanns, Raik Kuhlisch, Oliver Pfaff, Olaf Rode

Email: ihe@himss.org

CONTENTS

Introduction

The exchange of health data among enterprises is the subject of multiple IHE integration profiles. For example, XDS allows for sharing data among enterprises within an XDS Affinity Domain while XCA even enables the sharing of medical data across multiple of such domains or communities. As with any processing of personal data, various constraints (laws, regulations, and policies) apply to these data-sharing use cases. These regulations are driven by different aspects of medical data processing and therefore follow different objectives:

1. Protecting the patient's privacy and right to self-determination (e.g., HIPAA in the US and the European Privacy Directive)
2. Ensuring the integrity and proper handling of health data (e.g., regulations for the handling of radiologic data)
3. Enforcing an adequate risk management within organizations (e.g., KonTraG in Germany)

With respect to the prevention of inappropriate or illegal disclosure it is crucial that providers of medical data can be sure that data consuming parties enforce access constraints conformant to the purposes under which that data was provided. Therefore the definition and enforcement of access rules for medical data and services throughout clinical workflows is a precondition for any cooperative patient treatment. Perimeter protection (e.g., firewalls) and mutual node authentication (e.g., as provided by ATNA) are the foundation for any secure healthcare infrastructure. But with the establishment of XDS and other profiles – e.g., XDR, XDM, XCA – different groups of users are enabled to access a wide variety of medical information using different technical means and acting for different purposes. This diversity requires access control to be applied on data objects and workflows rather than on technical systems. In order to put the patient in control and align IT-security with enterprise-wide IT-governance there needs to be a shift of the definition of access rules up to the administrative and workflow semantics levels. At the same time the enforcement of these rules needs to move down into middleware layers.

1.1 Access Control Scenarios in Healthcare

Access control in healthcare has many requirements depending on the legal framework, the people and resources involved in a data processing scenario, and the weighting of security objectives within an enterprise. The complexity of medical workflows makes it impractical to analyze the complete set of use cases and scenarios that might occur.

The following list of simplified scenarios sketches the use cases that have been considered during the development of this white paper. This list is representative for the challenges of enforcing access control in healthcare. It is assumed that most real-life access control scenarios can be mapped onto these core use cases and methodologies provided by this white paper:

- Internal Resource Security: Within a hospital access to a patient's medical data is restricted to personnel who are involved with the patient's medical treatment and the corresponding administrative activities (e.g., billing). Access to certain sensitive information is further

limited to certain functional roles in order to ensure that this information is only disclosed to people who need to know it for a dedicated purpose.

- Patient Privacy Consent: Within a regional healthcare network the ability is provided to exchange medical patient data among the participating medical organizations (e.g., using IHE XDS). A patient may determine which organizations are allowed to access to their medical data.
- Secondary Use: A patient grants access to certain of his medical data to a medical study provided that all data is pseudonymized before use.
- Break Glass: In case of an emergency, access restrictions from patient provided policies and internal security regulations are overridden by a dedicated emergency policy which allows a physician to access all medical data of the patient. Part of this emergency policy is the obligation that a specific entry is written to a secure audit trail.
- Individual Opt-Out: A nurse is scheduled for a surgery in the hospital where she works. She does not want staff members working in the same department to get any insight into her administrative and medical data.

1.2 Scope of the White Paper

This document looks at the issues of how to define and implement access control in healthcare networks that might even span across communities. The focus is mainly on issues that relate to the IT architecture and the flow of messages that are required for a distributed access control scenario. Therefore this paper will deal with the problems of (1) how to apply established principles of secure design and SOA security on the design of access control systems and, (2) how to model an access control solution in a way that is well suited for reasoning and evaluation. It also begins the discussion of how to deploy an access control solution using well understood patterns and interoperable system components as seen in appendix C.

Given the strong focus on models and methodologies for designing access control solutions for cross-enterprise data exchange in healthcare, the primary intended audience are system architects and developers who are involved in the planning, design, and realization of regional healthcare networks and comparable infrastructures where the secure exchange of patient related data among enterprises is an issue.

The concepts presented in this paper are evolving rapidly and are subject to manifold national and international standardization efforts. The goal is to expose the common concepts from all of these activities, match them with experiences from existing healthcare networks, and define common design methodologies and technological building blocks which allow for a variety of strategies and policies to be used. The building blocks are described on a conceptual level and on an integration level based on current state-of-the-art in security token handling.

It is assumed that the design of the overall healthcare data exchange infrastructure is aligned to the principles of a service-oriented architecture (SOA). It is furthermore assumed that a dedicated security architecture is set up which provides a circle of trust among the security and business services which are deployed among independent XDS Affinity Domains. Nevertheless even if the focus is on cross-enterprise health information exchange (HIE) all concepts provided by this white paper can be scaled down to the organization or even department level.

Outline of the White Paper

The paper is organized as follows.

- Chapter 2 gives an overview of all recommendations that are made in the entire document. These start with high level generic recommendations, then shift into detailed recommendations taken from chapters 3 and 4, and then into recommendations for IHE gap filling taken from chapter 6. The detailed recommendations and gap filling may be somewhat confusing when read out of context. Their full context is introduced later in the document, where the recommendations can be found again. The glossary may be useful to understand the terms used in this chapter.
- Chapter 3 reviews fundamentals of access control, the state of the art, and introduces the Access Control System (ACS).
- Chapter 4 reviews policies and relevant attributes which are needed in order to design a proper policy-aware healthcare solution. It illustrates how different shapes of policy concerns might be harmonized and relevant attributes are bound to policies.
- Chapter 5 presents the methodology for designing an ACS by presenting an example use case.
- Chapter 6 fills the gaps in the current technical framework. There are specific recommendations for IHE activities. Some of what the model needs are already present in the IHE Technical Frameworks. There are other specific actors and transactions recommended for profiling, as well as some recommendations for educational whitepapers and recommendations against some other proposals that are likely to conflict with this recommended model and methodology.

1.3 Conclusion

There is no one-fits-all solution for access control in federated healthcare environments because the "optimized" deployment of policy administration and policy information – and the respective data flows - depend on the paradigms and the semantics of the policies used. There is no single international standard privacy policy. Privacy policies vary from location to location, and policies will vary over time.

Each individual federated healthcare environment must establish policies and the required information for access control decisions that match cultural and legal needs. Standards can then be used to enforce these policies. Therefore this white paper does not provide a single deployment of an access control system. Instead it provides healthcare system architects with a model and corresponding methodology that can then be used to enforce these different policies.

- The model describes how the building blocks of a distributed access control solution can be described in a manner that is both deployment and implementation independent.
- The methodology defines how system architects can use the model to reason about different realization opportunities in order to discover the most appropriate deployment and flow control with respect to the given requirements.

The model is illustrated in several difference use cases that correspond to different local needs. These illustrations and other analyses show that while a single one-fits-all solution can not work, this model and methodology does work with a variety of different requirements.

1.4 Open Issues and Questions

1. Define vocabulary for attributes reflecting use cases (e.g., in appendix)
2. What are the acceptable costs for an ACS? Other measures such as audit trails might be "cheaper" but they only react on misbehavior: they might solve the case, but the damage is done and cannot be repaired. Maybe we should add a section to chapter 3 with a short discussion on proactive vs. reactive and technical vs. administrative measures.
3. What about obligations and constraints? They should at least be mentioned.
4. Figure on behavior and access policies and on consent vs. compliance is confusing and must be redesigned
5. The use case is a simplified use case where the physician is shown performing accesses for historical data. This is often a delegated role, where the physician delegates a file clerk to perform this task. Should this added complexity be shown?

1.5 Closed Issues

1. Use of XACML terminology is acceptable because it is well established.
2. The phrase "Access Control System" will be used instead of "Access Control Service".
3. The implementation of delegation is defined rather than use of brokerage to manage trust delegation.
4. Please add glossary items for terms that should be defined.
5. The concept of context aware access control is introduced.
6. Just use cache mediation rather than define a policy cache.
7. Use policy override to solve policy conflicts.
8. The eCR example is moved to the appendix.

2 Summary of Recommendations

The following recommendations are made to people designing and deploying access control systems in health care IT environments as well as to IHE for further profile developments. The background and analysis for these recommendations can be found below. Each of these recommendations is marked to indicate the section where this analysis can be found, and identifies who the recommendations are for.

Recommendation (Section 4.1.1): *for policy planner and standard vocabulary developers*

Start small by defining coarse (rather basic) grained roles. Focus on functions and job titles rather than on concrete hierarchies. Use role refinement only for expanding permissions (more specialized roles should always have more rights). Keep in mind that there are more attributes for deciding on permissions than just the person's role. For example, if the types of reports a nurse might access, depend on the department she is working with, it may be easier to match attributes related to departments and report types, instead of refining the role of a nurse into dozens of sub-roles for each department. Think about using organizational measures and reactive measures (e.g., tracing audit trails) instead of technical restrictions, whenever suitable. Make sure that IT-compliance states clear rules for when to use which measures.

Recommendation (Section 4.1.2): *for application designer and architects*

Make software applications/systems explicit and clearly separate between them. State a clear purpose of use, consider typical all-day usage scenarios, and identify the tasks that relate to this purpose of use. Identify types of medical data objects and roles that process this data through the application/system. Consider the whole life-cycle of an application/system instance and its managed data. Determine the attributes that can be used to decide whether a certain identity might perform a certain operation on one of the managed objects. Make sure that there is an unambiguous mapping of roles derived from the enterprise's organization of work onto roles that are defined by the application/system.

Recommendation (Section 4.1.3): *for policy planner*

Consents should be policies that mainly control resource access by explicitly identifying authorized identities (individuals, organizations, groups). They should refer to a clear purpose of use or/and an organization's rules of governance (compliance) which determine how authorized personnel might process the patient's data. Whenever possible, patient privacy consents should grant access to organizations or groups instead of individuals, because this makes it easier to handle cases like temporary substitutions, reserve pool employees, and job rotation.

Recommendation (Section 4.2.2): *for policy planner and application architects*

Make sure that privacy consents always include an agreement of the patient to a clear purpose of use and the internal processes that are used to process his data. Make sure that especially software applications/systems shared among enterprises comply with all parties' rules of governance and that software application/system specific roles match with organizational roles. A chain of service calls is a service (i.e., service choreography). Make sure that the semantics of the top-level service is preserved throughout the whole chain. Always solve conflicts between consents and organization of labor on an administrative level.

Recommendation (Section 4.3.1): *for policy planner and application architects*

Clearly work out which attributes are required for the activation (selection) of a policy and for the decision on a policy. Consider attributes that are needed to decide on the implicitly defined meta-policy (e.g., override of all other policies by an emergency policy in case of an emergency access). Define the respective attributes stubs in a formal way; e.g., by mapping them onto data types or message fragments. Reason about the security status of these attributes by analyzing possible threats that might arise from corrupted attribute values. Define conditions that might require specific means to preserve the confidentiality, integrity, or authenticity of attribute values (e.g., transferring authenticated subject identifiers and role assignments over a public network).

Recommendation (Section 4.3.2): *for policy planner and application architects*

For each attribute stub that is required for either policy activation or policy decision, identify the respective attribute value sources. Identify which further attributes might be required to retrieve a certain attribute value from an attribute value source (e.g., retrieving a subject role will require providing a unique subject identifier). Define the respective attribute stubs and align them with previously defined stubs in order to limit the number of representations of attribute values (e.g., things are much easier, if a subject identifier needed for retrieving a role attribute is defined the same way as a subject identifier that is needed for deciding on a policy that is derived from a patient privacy consent). Verify, whether the analyzed level of security for attributes values can be provided by the attribute source or if it has to be provided by specific means within an ACS. Elaborate restrictions that hold for retrieving attribute values from an attribute source. E.g., an attribute service that provides role information might only be locally accessible or require a prior login which would place restrictions on the flow of control and on the deployment of functionality among the ACS.

Recommendation (Section 5.6.2): *for policy planner, application architects, standard profile developers*

Attribute values should always be pushed to the PDP, except for cases where

- the retrieval of an attribute's value requires information that is only available at the resource domain
- the context domain is unable to either discover an attribute's source or to establish a trust relationship with the attribute's source

If no mapping of attributes onto policies has been done at design time, the context side ACS components may not "know" which attribute's values are required for policy decision. In this case the context side ACS component should provide the PDP with all attributes they can gather (e.g., authenticated subject information, context identifier, patient identifier) and rely on the PDP to be able to pull all other attribute's values on demand.

Recommendation (Section 6.1): *for IHE planning and technical development*

IHE should define a framework for the definition of interoperable "get X-Assertion" and "provide X-assertion" transactions. This framework should consider two different levels of trust: direct trust (X-Service User consumes X-Assertion) and brokered trust (X-Service User as intermediary between X-Service Provider and Security Token Provider).

Recommendation (Section 6.1): *for IHE planning and technical development*

IHE should provide guidance on how XUA can be used in cross-domain scenarios based on XDS and XCA without having to bridge or connect both domains' PKIs.

NOTE: The European epSOS project will specify such a solution with certain gateways (national contact points) acting as guarantors.

Recommendation (Section 6.2): *for IHE planning and technical development*

IHE should define an attribute provider (semantic of a policy information point) for querying attributes about objects (i.e., subjects, patients, and resources). The respective actors and transactions are needed for infrastructures where e.g., subject authentication is performed within a domain that does not maintain role and organization membership information about the authenticated subject (e.g., if a health professional card is used) authentication is performed within a central subject domain (e.g., a nationwide PKI) while most of the subject's attributes are managed with the enterprises subject domain (e.g., enterprise HR services).

Recommendation (Section 6.4): *for IHE planning and technical development*

IHE should provide guidance on how to use XDS stored queries and existing document retrieval transactions to implement dedicated transactions for policy activation (claim retrieval) and policy retrieval. For direct trust scenarios such guidance is already given in the BPPC profile, but this does not cover brokered trust scenarios.

Recommendation (Section 6.5): *for IHE planning*

IHE should not define specific means for explicit subject role activation. An implicit role activation should be preferred because this can easily be implemented by using existing standards and profiles.

Recommendation (Section 6.6): *for IHE planning and technical development*

No additional actors/transactions are needed for single XDS Affinity Domain scenarios. For multiple XDS Affinity Domain scenarios the interoperability issues can be reduced to PEP-PDP communication and the syntax of encoded policies. Due to a close integration of these building blocks by existing products there seems to be no urgent demand for normalization. Therefore IHE should take a pragmatic approach and provide a white paper or cookbook on how to integrate a PEP/PDP into the XDS flow of transactions assuming a close integration of PEP and PDP.

3 Fundamentals of Access Control

Providing complete and effective security allows no room for mistakes. Therefore systems designers must stay on the safe side by applying and evolving existing concepts rather than inventing new security mechanisms from scratch.

3.1 State of the Art

In this section the state-of-the-art with respect to access control is described. All of the concepts provided in this section have evolved over the last 20 years and are well established. Their consideration in access control design therefore heavily contributes to the security and implementability of the overall system.

3.1.1 Principles of Secure Design

Security cannot be added to an existing system; it must be built into a system as part of the initial design. The following list sums up some of the most important design principles for access

control solutions that have evolved over the last 20 years[1]. The concepts in square brackets after each list item refer to points covered by this white paper which considered the respective principle.

- Economy of Mechanism: The mechanism used must be kept as simple as possible. It should also target a well-confined issue instead of trying to cover each and every eventuality. With respect to access control in healthcare one should always consider whether technically driven access control, organizational rules, or reactive measures - e.g., logging - are most appropriate for the implementation of different access restrictions. [need-to-know principle; use of policy templates]
- Complete Mediation: Every access attempt must be explicitly safe-guarded through the access control mechanisms at all times. This also implies that there must be no possibility to bypass access control (even for special roles/functions such as administrators) while accessing any resource. [policy based access control (PEP/PDP), emergency policy, patient safety]
- Open Design: All algorithms and security mechanisms have to be openly available and fully verifiable [standards].
- Least-Common Mechanism: Mechanisms, system states and objects, which are capable of explicitly or implicitly granting access rights, should not be shared between different users and/or software applications. For each individual user or software application, a different mechanism or an instantiation of the same mechanism should be used.
- Fail-Safe Defaults: Anything which is not explicitly allowed is denied by default. That means in practice, that any access attempt, which cannot be explicitly and entirely verified as allowed, is automatically denied. Privileges are admitted exclusively to an initially empty set of rights (opt-in in favor of opt-out rules). [needs-to-know principle, patient safety]
- Separation of Privilege: Whenever possible, security and safe-guarding mechanisms should verify multiple, independent conditions before granting access to a protected resource (e.g., a user must match a certain role and issue the request from a trusted system). Well known concepts based on this principle are “separation of duty” and “two-man rule”.
- Least Privilege: Any identity should be granted with the least set of access rights possible in order to complete its assigned function or task. If any right needs to be added to perform extraordinary duties, the right may be granted when actually required and discarded after the completion of the non-standard task [role engineering, role activation].

[1] The principles of secure design considered in this white paper are consolidation of several sources:
- Saltzer, J. H.; Schroeder, M. D.: The Protection of Information in Computer Systems. Proceedings of the ACM. Vol. 63, Nr. 9. pp. 1278-1308. 1975.
- Wallach, D.; Balfanz, D.; Dean, D.; Felten, E.: Extensible Security Architectures for Java. In: Proc. 16th Symposium on Operating Systems Principles, October 1997, Saint-Malo, France.
- Benantar, M. (Ed.): Access Control Systems. Springer. 2006.
- Bishop, M.. Computer Security: Art and Science. Boston, MA: Addison-Wesley, 2003
- Ferraiolo, D.; Kuhn, R.; Chandramouli, R.: Role-Based Access Control. Artech House. 2. Edition, 2007.
- Build Security In. Website of the US Department of Homeland Security. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/principles.html

- User Acceptability: Mechanisms securing and safe-guarding access to a resource must not add unnecessary burden to the user. Excess burden leads to lack of acceptance which may create situations where users find creative ways to bypass many of the other design principles (e.g., everyone in a hospital department uses the most privileged account).
- Reluctance to Trust: Every system, human actor, and runtime environment should always be considered as potentially insecure. Trust should never be loosely given. Furthermore, it is imperative to limit and minimize the amount of mechanisms, systems, and objects, which are required to be fully trusted required by the access control subsystem [client-side vs. resource side enforcement, circle of trust].
- Isolation: All access control/management mechanisms should be isolated from other systems, operated independently, and must be specially secured [policy based access control (PEP/PDP), SOA design principles].

3.1.2 Common Access Control Models

The four fundamental access control paradigms[2], which are found and distinguished in practice today, are:

- Discretionary Access Control (DAC): The validation and grant of access permissions is solely performed on the basis of the concrete identity of a subject and its group membership. Subjects who possess access rights for a certain resource may pass those on to other subjects. In a quite common implementation of DAC (such as used in UNIX), to each resource a special property called the "owner" is assigned, who may exclusively grant or deny any access rights to users or other groups for this resource.
- Mandatory Access Control (MAC): The validation and grant of access rights is performed by utilizing sensitivity levels/labels, rules, and/or policies. In the simplest case each protected resource and each user possesses a set of security attributes: the user is usually assigned a clearance level, whereas the resource is assigned to a sensitivity level. Rules determine how those levels may concretely correlate and whether and how users of one clearance level may grant access rights to other users of different clearance levels. Depending on these rules different security objectives can be supported; e.g., if confidentiality is the major objective the Bell-LaPadula Model is more appropriate while integrity protection is better supported by the Biba model. In many running systems further dimensions – e.g., a specialty – are added for dealing with equivalency classes of sensitivity and/or clearance levels (lattice-model).
- Role-Based Access Control (RBAC): The RBAC-model is not assigning access rights to any resource directly to a subject's identity. Instead, each subject's identity is assigned with a set of roles, in which any access rights are defined. The concrete executing of access rights is therefore not directly bound to the user but acquired through its current role. The roles and its concrete permissions may be defined hierarchically (hierarchical RBAC) and rules may be

[2] The four models sketched here are the ones that have been considered with existing healthcare standards and that are well suited to cover all of the use cases sketched in section 1.1. Nevertheless for certain scenarios other models and theories (e. g. Clark-Wilson or Take-Grant) might be more appropriate, especially if specific integrity requirements have to be met.

constructed, which defines limitations (constraints) for the role assignment and permissions-granting (constrained RBAC).

- Context-Aware Access Control: The shift from DAC and MAC to RBAC come along with a decoupling of subject related issues (identities and roles) and resource related issues (permissions encoded within policy sets). Context-aware access control goes even a step further by breaking up the static assignment of identities to roles and the static assignment of policies to roles or resources. It does so mainly by providing additional rules that control these assignments and as such introduces a new level of indirection.

Many countries' laws consider the patient as the sovereign of his medical data (my body - my data - my control). Therefore the patient implicitly is the owner of his data and the only one entitled to grant access permissions. This leads to a DAC influence on every access control solution in healthcare. This influence is rather implicit in an intra-enterprise scenario where a treatment contract is usually signed by the patient which transfers part of the patient's rights to the hospital (which is as well part of the DAC paradigm). In an inter-enterprise or cross-domain scenario this DAC-portion of the access policy is much more visible and usually expressed by a patient's privacy consent.

MAC requires subject-classes and object-classes to be partially ordered. Assuming that functional roles are used for permission assignment, this partial order must be either coarse-grained or artificially constructed. IHE BPPC is suitable for implementing rather compact MAC style access control features using HL7 confidentiality codes which are e.g., supported by IHE XDS, DICOM, and HL7.

RBAC is best suited to align access permissions with the organization of labor within a medical organization. Especially for intra-enterprise access control RBAC is evolving to be the paradigm of choice. For the last years VA and HL7 have provided guidance on the process of permission definition and on adaptable catalogs of roles and permissions. These efforts and the underlying "needs-to-know" principle will be discussed in more detail in chapter 4.

Context-aware access control is well suited to overcome some of the inflexibilities of RBAC in healthcare where people often switch among multiple roles, take over roles of others in specific situations, and where access rights vary depending on the state of the patient or the "operational mode" of the organization (e.g., nightshift, disaster management). In this paper core aspects of context-aware access control are considered by defining the activation of roles and policies as (potentially rule-based) functionalities rather than as static assignments.

3.1.3 Policy Based Access Control

All commercial implementations of the above named and described access control models are based upon the strict separation - as illustrated in the figure below - of policy enforcement and policy decision.

- A policy is considered to be a set of rules, which control the security and privacy behavior of a given system. The security policies addressed in this paper define what subjects are authorized to access what objects for which purpose.
- A Policy Enforcement Point (PEP) as a part of the access control system intercepts all access attempts for any protected resources. For this, the PEP sends an authorization decision

request to a PDP. Afterwards, it receives the authorization decision from the PDP and enforces its result and associated obligations.

- A Policy Decision Point (PDP) may decide on the concrete outcome of an incoming authorization request. In order to do that, the PDP applies the best-fitting policy and may determine additional security attributes when necessary. The PDP then evaluates the applicable policy in the context of the incoming authorization request. The result is coded in the form of a policy decision, determining whether the requested access operation is granted or denied.

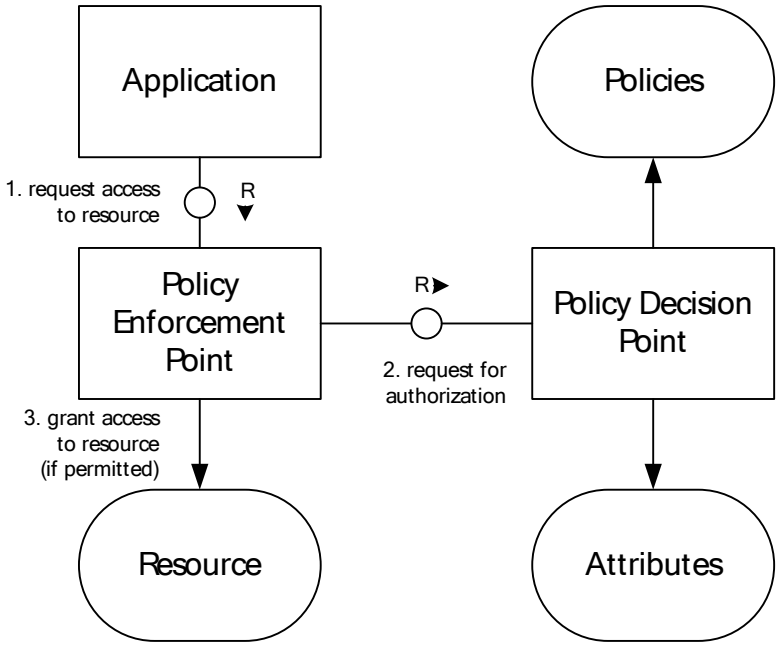


Figure 1: Policy Enforcement and Policy Decision

XACML defines two further actors:

- The Policy Administration Point (PAP) who basically administers and maintains the policies.
- The Policy Information Point (PIP) which facilitates the PDP in acquiring any additional security attributes of resources and subjects in order to determine whether an access request is to be granted or denied.

3.2 SOA Security Principles

Sharing medical resources in an automated fashion mandates authorization, authorization calls for authentication, and authentication depends on identifiers as well as credentials. This basic formula holds for IT-systems based on new architectural approaches such as SOA and Web services as well as for traditional systems in enterprise IT. While new architectural approaches do not change this formula, they impact how security should be realized. Separating clearly between the supply of security functionality in the security subsystem and its use in business services is important to avoid duplication of work, foster interoperability, and facilitate re-use.

Hence, security tasks such as identification, authentication, and authorization are externalized from business services.[3]

Decoupling the tasks of authorization and authentication leads to a model where federated protocol endpoints are used to transfer authentication events in a trustworthy fashion. The standard approach is to have a normal authentication process, which results in a transferable representation of authenticated subject information that can be consumed by the authorization side. An appropriate mechanism for exchanging authentication information this way is provided by the IHE XUA interoperability profile which is based on the SAML standard.

Decoupling authorization and authentication is about separating the use of authenticated subject information from its construction. The abstraction of a Security Token Service (STS) provides the core mechanism for this architectural principle. Security Token Services are actors (services) that are dedicated to the processing of security tokens such as SAML assertions. Security tokens are not restricted to subject authentication, they can as well be used for decoupling the issuance and consumption of policies, policy decisions, and all kinds of attributes that are needed for the evaluation of a policy.

Security Token Services are not confined to serve only their local domains but they can also be used to broker trust among domains. Trust brokering is used whenever a party (party A) has to rely on another party's (party B) conformance to a certain policy or behavior but party B is not able to express this compliance in a way that party A would accept as trustworthy. In this case a third party (the broker) is used which is "known" as trustworthy by party A and which provides kind of a guarantee for certain claims party B states about itself. By using security token services which are "known" across multiple domains (that is their certificate - assigned to a common understanding of the hereby expressed guarantees - can be verified across domain boundaries and) these domains can be federated in a way that security token issued in one domain can be consumed by actors within another domain.

With respect to access control the concepts of brokering and delegation must not be mixed up. While trust can be brokered, it cannot be delegated. What can be delegated are access rights (e.g., a physician delegates certain of his rights to a nurse to enable her to perform certain actions on his behalf). Both concepts can be combined; e. g. by using trust brokering to transfer claims on the legitimacy of a delegation to an actor that for himself is not able to verify this legitimacy.

3.3 Access Control System (ACS)

Following the principles sketched above all security related tasks should be provided by a dedicated security architecture which is decoupled from the business architecture. Within the security architecture different subsystems are designated to different security services such as authentication, authorization, non-repudiation, etc. An Access Control System (ACS) is a (logical) capsule around all authorization subsystem components that are (logically) linked with an actor.

[3] Moreover, SOA security also includes services for confidentiality and integrity, availability, auditing and compliance. Since this whitepaper puts the focus on access control, these security terms are partially neglected.

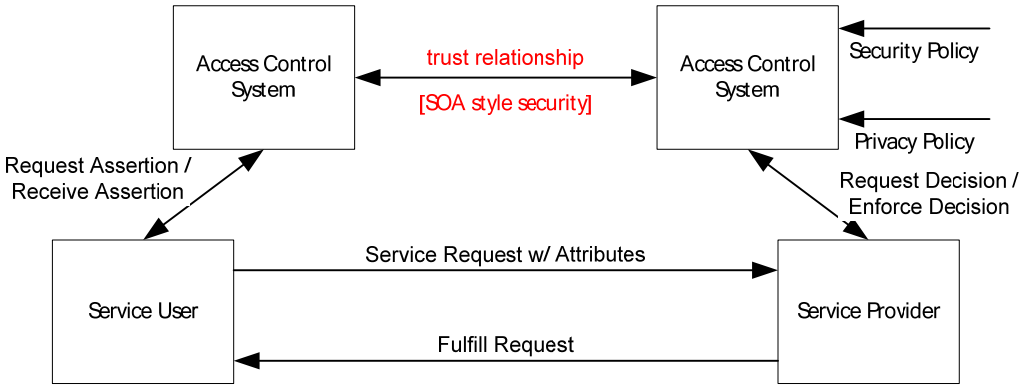


Figure 2: Actors with loosely coupled Access Control Systems

As shown in Figure 2 each actor that consumes or provides business functionality is bound to its local access control system. While the business services always communicate directly through request and response messages, their respective ACS might only be loosely coupled. E.g., a consuming service might request assertions on the user's identity and role from its local ACS and sends them to the service provider as part of the request message. The service provider calls his local ACS for the decision on the acceptance of the request. Part of this call includes passing the assertions that were received from the consuming business service.

3.3.1 Building Blocks

Each access control system makes use of several logical actors: PEP, PDP, PIP, and PAP. Each of these actors might appear in different instances (e.g., PIP for the retrieval of subject attributes vs. PIP for the retrieval of resource attributes) that could originate in different logical domains.

There is no fixed deployment of these actors among the participating systems. A common ACS therefore is able to integrate components for the

- Management and Retrieval of policies (PAP)
- Management and Retrieval of attributes (PIP)
- Policy decision and policy enforcement (PDP/PDP)
- Security token issuing and verification (STS).

Figure 3 shows the logical building blocks of a common ACS that correspond to these technical components. The PIP and PAP actors are represented by Attribute Services and Policy Authorities.

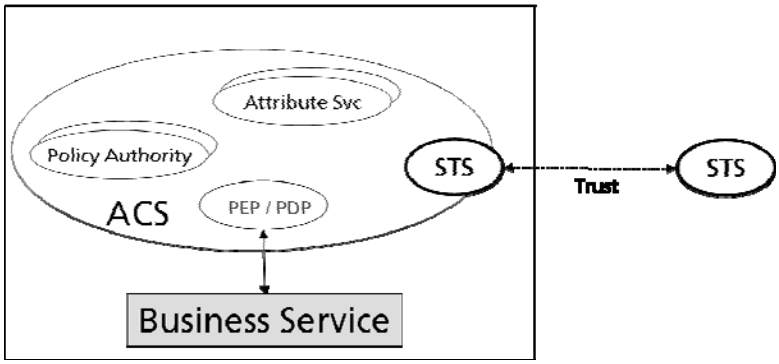


Figure 3: Building Blocks of an Access Control System

These building blocks – which will be discussed in depth throughout the rest of this white paper - are:

- Policy authority services for the management, selection, and activation of policies.
- Attribute services for managing attributes an subjects, resources, etc. which have to be considered for the evaluation of policy rules
- Security token services for the establishment of trust relationships to remote ACS and for the secure (with respect to integrity and authenticity) exchange of attributes and policies across domains.
- Policy enforcement and decision points which apply policies on business service requests. It is assumed that (at least on a model level) each request is mediated through the PEPs of the communicating nodes.

3.3.2 Access Control Domains

In order to design a proper access control system, it is helpful to use a generic model that supports the discussion and evaluation process. This model must be independent from the deployment of business services and the distribution of ACS components among physical nodes. For this, a model of logical access control domains is introduced where each domain considers one specific access control aspect in the execution of a business transaction.

In the simplest case three domains are modeled: context domain, subject domain, and resource domain. Each domain's ACS is derived from the common ACS model sketched above:

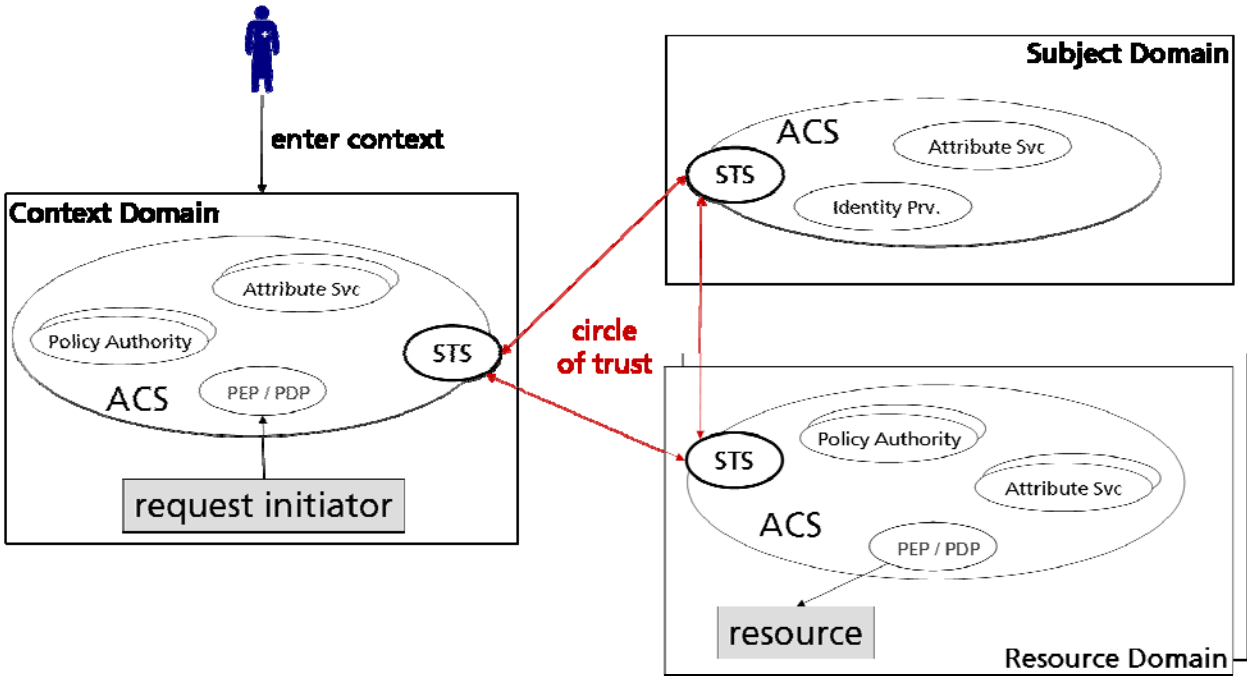


Figure 4: Core Access Control Domains

The issuer of a request affecting a protected resource is located within the **Context Domain**. Since the service consumer is located at this domain, the assertion and message flow is controlled within that domain, too. The context is implicitly selected by entering it (e.g., opening a form in a hospital information system).

The context domain provides an attribute service for managing attributes (e.g., current step in a medical workflow) and a policy authority for local policies. By entering the context, specific roles of the subject might be selected which are served by the subject domain's ACS. In addition, a PEP/PDP for the enforcement of local policies (e.g., for constraints on role activation) resides here. Details on those concepts can be found in subsequent sections.

Authoritative identity sources with manageable subject attributes accomplish the basis for identity services in the **Subject Domain**. The subject domain hosts the identity provider (a dedicated Security Token Service) which is responsible for identifying and authenticating subjects by their claims, and encapsulating the respective assertions such that other STS can validate them. An additional attribute service renders subject attributes from directory services or other information systems.

The **Resource Domain** is characterized by the management of protected resources (e.g., medical database or EHR). Resources are marked with attributes that are considered as metadata. In order to restrict the access to protected resources, security and privacy policies might be rendered and evaluated by a resident policy decision point.

The linkage of the resource consumer and provider with their respective nodes' PEP denotes that a complete mediation is enforced by either intercepting the access operation at the consumer's or the provider's side (or even at both sides) in order to decide on its acceptance with respect to the active policies.

3.3.3 Federated Healthcare Environments

The big challenge in healthcare environments is effective cooperation. Cooperating hospitals contribute to improved quality, strengthening of responsibilities, and decrease of costs. In general, a lot of use cases came up in the context of integrated care in national or international eHealth initiatives. A special characteristic of many scenarios is that cooperation among the participating enterprises is driven by on-demand, patient-group-specific, or disease specific issues. E.g., while hospital A might run a regional eye-care-network with hospitals B and C, it might co-operate with hospital B and nursing home D for the treatment of certain mental diseases. Other partners might join these networks on demand.

Dynamic networks like these where each partner is in control of its own resources and where mutual trust – e.g., in authenticated identities of other partners - is established when needed, are called federations. By their ability to broker trust, the participating partners form a "circle of trust" that allows for a security token to be issued and consumed by all partners.

With respect to health information exchange, two major federation scenarios have to be considered:

- Federation of resource domains: A context domain is able to request protected resources from multiple resource domains. Therefore the ACS at the context domain has to be able to share its attributes and policies with ACSs within multiple resource domains. This includes the ability of all attribute managing domains to provide attribute values in a way that can be understood and recognized as authentic by the domain that decides on the policy.
- Federation of subject domains: A resource domain accepts requests from subjects that reside in different subject domains. By federating identities, subject domains are able to discover, exchange, and synchronize authenticated subject attributes that relate to the same subject.

Usually both scenarios are given at a time, e.g., in a peer-to-peer network of co-operating hospitals, both subjects (identities) and resources might be federated in order to give requestors the impression to act on a single data base. ACS that communicate among each other through security tokens are implicitly federated and therefore best suited to provide access control even for highly distributed and ad-hoc established cooperation models.

3.3.4 IHE Profiles

Many of the ACS building blocks are already covered by existing IHE profiles:

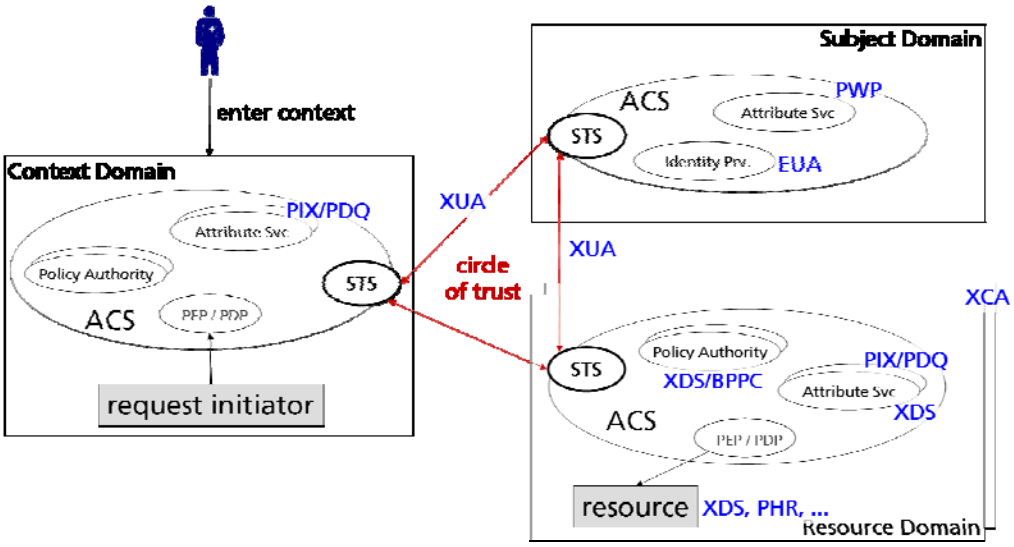


Figure 5: IHE Profile Mapping

The Cross-Enterprise User Assertion (XUA) integration profile provides mechanisms to exchange authenticated subject information across domain boundaries. It is therefore well suited for connecting the subject domain to the circle of trust. By combining XUA and Kerberos-based Enterprise User Authentication (EUA) an already IHE compliant enterprise can run its own identity provider using existing technology. The Personnel White Pages (PWP) integration profile defines how organizations might maintain attributes describing their personnel based on common LDAP technology. By either integrating this information into an XUA assertion or by providing it to external users through security tokens (e.g., using an STS-safeguarded policy information point) the subject domain's required functionality can be provided by already existing IHE profiles.

The management and provisioning of attributes on patients is subject to the PIX and PDQ integration profiles. Depending on the deployment, the respective attribute services can either be located with the context domain, the resource domain, or a dedicated patient domain (see figure 5).

The Basic Patient Privacy Consent (BPPC) integration profile describes how XDS registries and repositories can be used for maintaining privacy policies. This allows for setting up a policy authority within the resource domain. By encapsulating this functionality by a security token service, privacy policies can even be exchanged in a secure manner across domain boundaries.

XDS registries are designated for the management and provisioning of resource attributes and as such provides the functionality of an attribute service. Using existing profiles for the management of policies and resource attributes at the resource domain and for the trusted exchange of subject attributes among domains, even rather complex access control scenarios can

be implemented. The major gaps not covered by IHE integration profiles are PEP/PDP and policy authorities which are decoupled from the resource managing systems.

4 Policies and Attributes

Each system can be described by its actors and the transactions that exchange data among these actors. Aside from the message payloads for authorization requests and responses, access control related data exchanged between actors of ACS within different domains are mainly policies and attributes. Which concrete transactions and actors are needed in order to set up a specific access control solution for a health information network depends on the amount and characteristics of the policies and attributes that are required to fulfill determined protection requirements.

The access demands derived from a medical treatment (Context/Purpose) must be in accordance with the potential permissions granted to a subject (Who). These permissions are based upon job function, organization of labor, etc. (Role, When, Where). The context-specific intersection illustrated in figure 6 reflects on the one side the resulting permission set of a subject on a technical point of view; on the other side, it relates to the information this subject "needs-to-know" in order to fulfill necessary tasks.

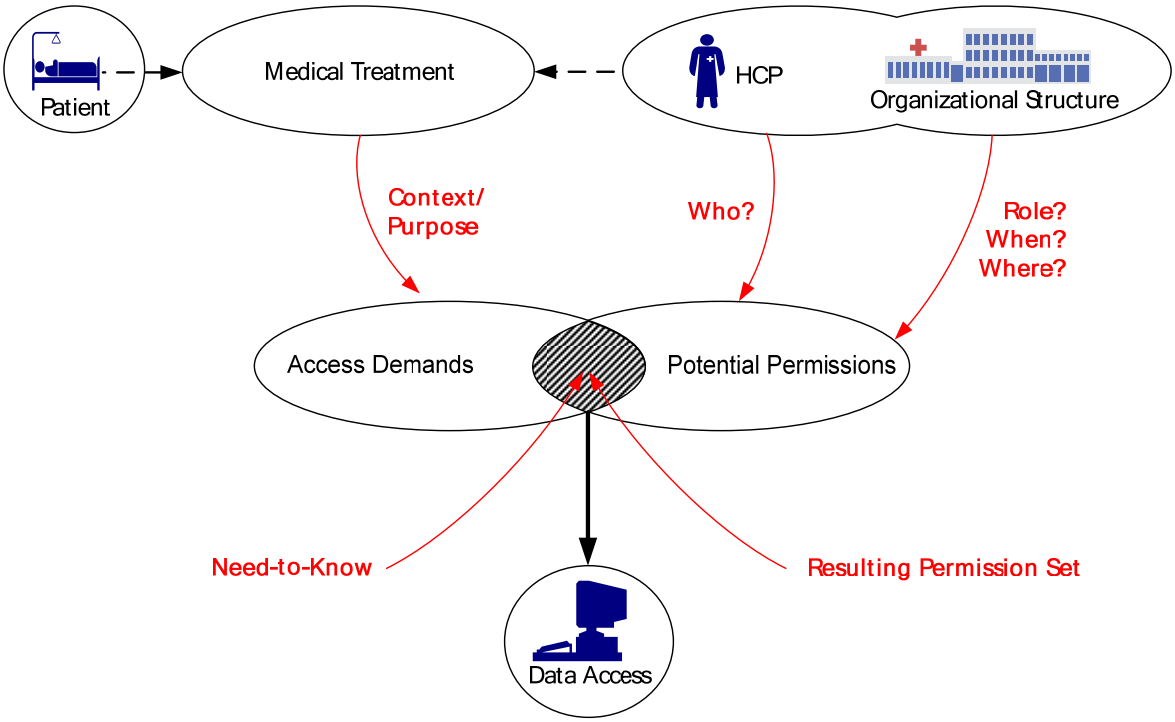


Figure 6: Derivation of Rights by the "Needs-to-Know"-Principle

4.1 Separation of Policy Concerns

Access control policies determine who is allowed to access which resources for what purpose within what contexts. Generally, a policy consists of:

- Target identifiers that denote which (kinds of) entities (resources, patients, subjects, etc.) are protected by the policy
- Conditions, that define rules for applicable targets
- Rules and constraints for deciding on whether a certain permission is granted or not
- Obligations that have to be processed in order to fulfill requirements that are associated with certain permissions

Policies might be nested in order to realize object-oriented concepts like specialization and inheritance on the various building blocks.

Given the generic and flexible organization of a policy[4], policies can be used to express nearly any constraint related to the processing of medical data. Given the multiple sources (e.g., legal, regulations, patient consents) for these constraints, one single policy for expressing each constraint may be very complex and specific. Therefore, it is very advisable to separate policy concerns and define different policies for different concerns that can then be combined accordingly on demand.

With respect to healthcare information exchange, three major concerns need to be considered:

- Patient privacy consents: constraints the patient puts on the use of his data
- Purpose of use: constraints derived from the intended use of a certain healthcare system that mediates (or even initiates) access to a protected resource
- Compliance: resource security rules for protecting medical data within an organization from illicit disclosure

Figure 7 illustrates the separation of policy concerns on the example of an electronic health record (EHR). It also shows how these concerns relate to each other and in what ways they correspond to policies that have to be enforced in conjunction with the processing of medical data.

[4] How much of this flexibility is really usable depends on the expressiveness of the policy language used.

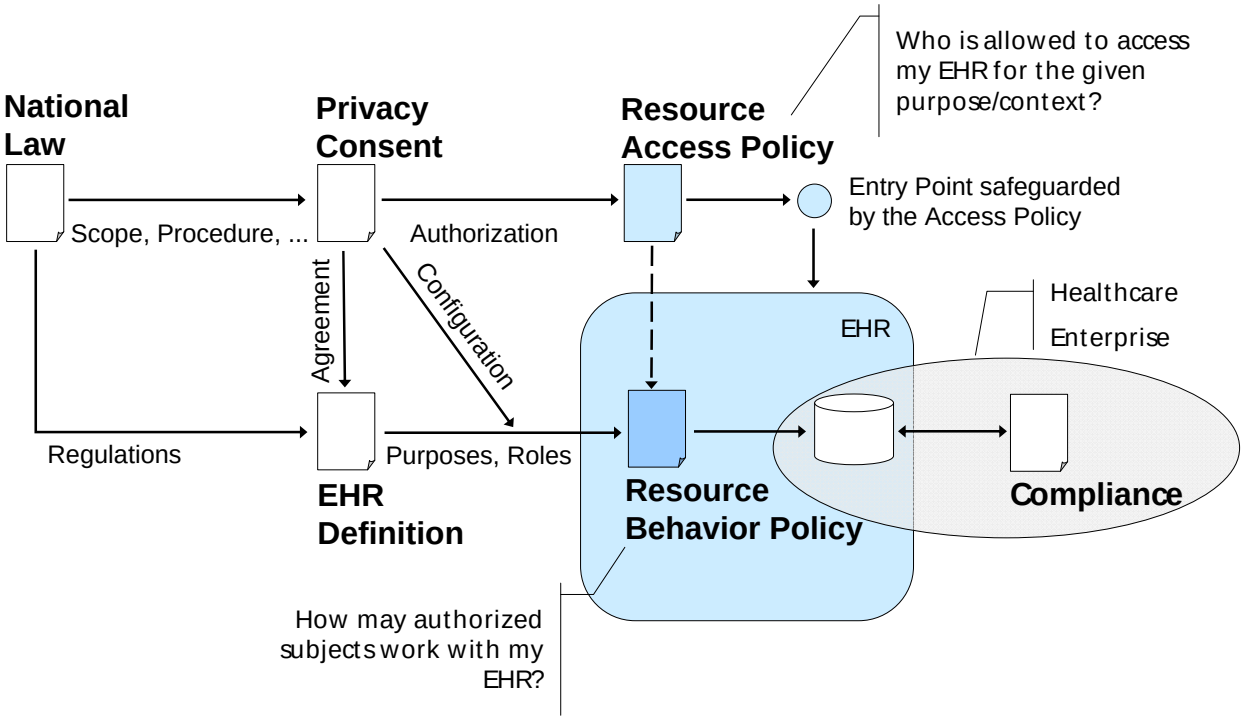


Figure 7: Separation of Policy Concerns

The initial baseline that is always considered is the national law. It determines the scope and procedures for patient consents and provides regulations on the framing conditions for the use of healthcare systems such as electronic health records (EHR).

Access to medical resources is always mediated through healthcare systems such as a health record. Each system has its specific purposes of use, functional roles, flow control, content constraints, etc. A patient **privacy consent** should always be based on the purpose of use of such a system.

The purpose of use can be translated into a **resource behavior policy.** This policy defines how certain subjects might act on certain objects that are managed by the healthcare system. Depending on the purpose of use and its implications, the patient might even influence the permissions expressed by this policy (configuration). For instance, the patient restricts the types of data that might be processed by the healthcare system.

A resource is always managed within the context of an organization (e.g., a hospital) that is liable for the lawful processing of their patient's data. It is the responsibility of IT-**compliance** to define roles, permissions, and obligations for internal and external data communication. The respective resource behavior policies are enforced whenever an access to an internal resource is requested.

Another major part of patient privacy consent is the authorization of certain individuals, organizations, and/or rules to use the healthcare system with respect to the agreed purpose of use. Translated into a policy, this part of the consent may be characterized as the **resource access policy.** This policy controls who is able to access a protected resource through the entry point (point of service application) of a healthcare system. The notion of an entry point is especially

755 important, if there are multiple of them (e. g. one entry point for medical staff and one for system administrators) that are safeguarded by different policies that define different expectations on the objectives and behavior of the respective user groups.

An access policy always corresponds to an associated behavior policy. For instance, after access to an entry point is permitted by a resource access policy, all resource operations are controlled
760 by the permissions expressed through the resource behavior policy.

4.1.1 Compliance: Resource Security

The policy concerns of an enterprise's IT compliance is quite similar to the concerns of the purpose of use since it clearly is a source for roles, tasks, and authorizations derived from restrictions on role-task assignments:

765 • The medical business activities of the enterprise are defined by tasks and scenarios, which in turn determine the purposes of use for medical data processing activities
• The enterprise defines roles – assigned to tasks - that reflect the enterprise's organization of labor

Compliance ensures that the assignment of roles to tasks and the assignment of people to roles
770 are fully aligned to the legal framing and the rules of governance of the enterprise. Behavior policies for resource protection can directly be derived from the organization of labor:

Everybody, who is allowed to perform a certain task, must also be allowed to perform all data processing that is required for performing this task. E.g., everybody who is allowed to do a surgery must be granted permissions to read relevant examination reports and to write a surgery
775 report.

Baseline for resource security that follows this **need-to-know principle** is a collection of roles and assigned permissions through a role-engineering process. Healthcare organizations, such as HL7 and VA, propose a scenario-based approach. In this approach typical procedures are excerpts of medical actors that are illustrated and described in a narrative. Each step in a scenario
780 incorporates the operations that are executed onto the medical or administrative objects. In order to successfully perform those operations, the required permissions are combined into catalogues and assigned to profiles. Inversely, scenarios are combined to tasks on a higher, conceptual level.

The outcome is a structured catalogue that illustrates what permissions (operations on resources) are needed in order to fulfill the particular scenarios. In a second step, the identified actors are
785 integrated, creating a matrix manifesting the roles and their required permissions.[5]

To each subject of a healthcare enterprise several (one or many) roles may be assigned, depending on the current work context and the daily schedule. In order to follow the design

[5] The HL7 security technical committee specified a detailed permission catalog for healthcare environments using role based access control. Core RBAC elements (users, roles, objects, operations, and permissions) are transferred into operation and object definitions that can be adopted. Moreover, the catalog appoints non-normative permissions that can be assigned to healthcare personnel. [Reference: RBAC Healthcare Permission Catalog, v3.38. November 2007. HL7 Security Technical Committee.]

principle of least privilege, the ACS must ensure that each person's current roles are only those roles that correspond to this person's current (medical) activities.

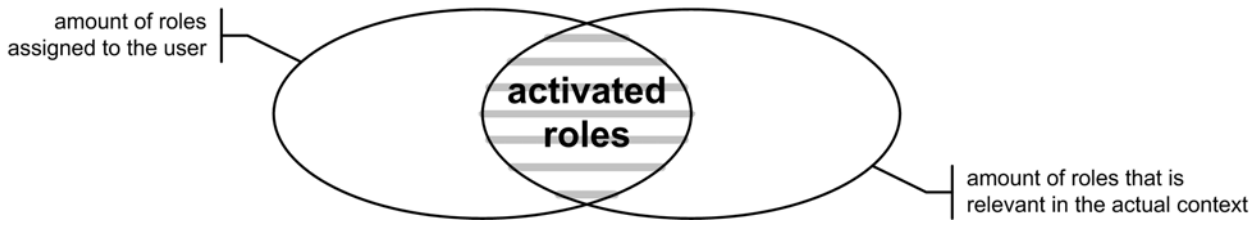


Figure 8: Role Activation

The current roles are determined by calculating the intersection of the user's theoretically assignable roles (all roles administrated for him in the subject domain) and the roles required to act in the current context. The activation (identification) of the current context is usually an implicit side effect caused by actions such as switching software applications and dialogs. Such an action might be when an administrative person of a hospital opens the "admission" software application or when he selects the "admission" dialog in the hospital information system. Then the current context is "patient admission" which might lead to an activation of this person's "admission personnel" role.

Recommendation: Start small by defining coarse (rather basic) grained roles. Focus on functions and job titles rather than on concrete hierarchies. Use role refinement only for expanding permissions (more specialized roles should always have more rights). Keep in mind that there are more attributes for deciding on permissions than just the person's role. E.g., if the types of reports a nurse might access, depend on the department she is working with, it may be easier to match attributes related to departments and report types, instead of refining the role of a nurse into dozens of sub-roles for each department. Think about using organizational measures and reactive measures (e.g., tracing audit trails) instead of technical restrictions, whenever suitable. Make sure that IT-compliance states clear rules for when to use which measures.

4.1.2 Purpose of Use and Policies

The purpose of use determines the answer to questions such as:

- What tasks can be performed using the underlying software application/systems? What are the scenarios where these tasks are performed?
- Which tasks are performed by the same groups of users? How can these groups be characterized?
- What data is processed by the software application/system? What operations are defined on this data?

In the previous section the need-to-know principle was introduced as a n:m relationship between a subject (e.g., a physician) and a protected resource. As any access to protected resources is

mediated through software applications/systems, this single "logical" relationship is split into two "physical" relationships: A "need-to-use" relationship and a "mediates access" relationship (Figure 9).

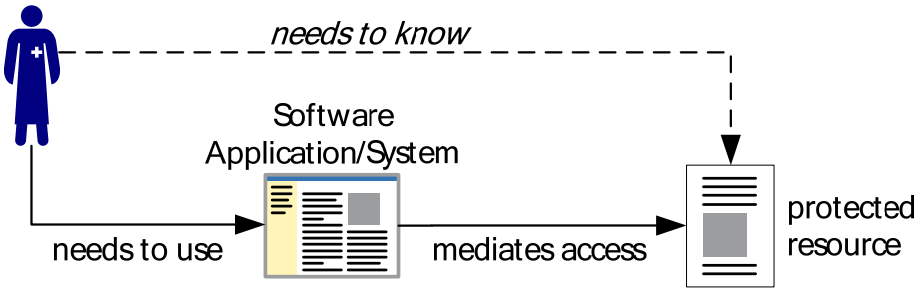


Figure 9: "Need-to-use" and Mediation of Access

A proper software application/system design has to ensure that the set of all valid sequences of a "need-to-use" and "mediates access" relationship is semantically identical with all valid "need-to-know" relationships that reflect the purpose of use.

Recommendation: Make software applications/systems explicit and clearly separate between them. State a clear purpose of use, consider typical all-day usage scenarios, and identify the tasks that relate to this purpose of use. Identify types of medical data objects and roles that process this data through the application/system (e.g., by using the role engineering methodology that was sketched in the previous section). Consider the whole life-cycle of an application/system instance and its managed data. Determine the attributes that can be used to decide whether a certain identity might perform a certain operation on one of the managed objects. Make sure that there is an unambiguous mapping of roles derived from the enterprise's organization of work onto roles that are defined by the application/system.

4.1.3 Patient Privacy Policies

In order to lawfully collect, store, process, and communicate medical information about a patient, concrete and prior authorization for those operations is required. This authorization is also referred to as the patient consent. This consent is the result of a patient's independent and informed decision and is specifically defining:

- What of his medical data may be shared
- For what purposes
- To what extent (partly, context-dependent, all)
- With whom (identities or organizations)
- For how long?

Finding a suitable technical representation of the patient's consent is considered to be quite challenging due to the potentially high complexity of an adequate reflection of the concrete patient's wishes.

Furthermore, the patient holds the right to withdraw his consent at any time, even during the treatment. Therefore opportunities must be provided that allow medical staff to determine

whether the consent is currently valid. The specific rules and regulations, which are encoded in the consent, directly reflect all explicit and implicit authorizations that may result from the patient's decisions. Simply spoken; a privacy consent expresses the patient's choice on whom he trusts and whom he does not.

While purpose of use and compliance related policies mainly control how authorized users process protected resources, a privacy consent usually focuses on determining who is allowed to access the resource.

A common instance for this separation of policy concerns is a treatment contract between a hospital and a patient: By signing the contract the patient

- Grants access to his data to all hospital employees who are involved with his case (admission, medical treatment, reimbursement, etc.)
- Agrees to the hospitals compliance which determines how, when, and for what reason hospital staff members can access his data

In practice, the patient just expresses that he trusts the hospitals to act responsibly and compliantly with his data. However, given the context (patient seeks medical help), the complexity of a hospital's organization of labor, and the diversity of roles / data objects, this is all the patient can do.

Recommendation: Consents should be policies that mainly control resource access by explicitly identifying authorized identities (individuals, organizations, groups). They should refer to a clear purpose of use or/and an organization's rules of governance (compliance) which determine how authorized personnel might process the patient's data. Whenever possible, patient privacy consents should grant access to organizations or groups instead of individuals, because this makes it easier to handle cases like temporary substitutions, reserve pool employees, and job rotation.

Consents might occur in different shapes, ranging from inferences from the patient's acts to a formal contract. A specific form of inferring permissions from a patient's acts can be implemented using electronic health cards or other patient-bound tokens.

Cards are a common mean to complement traditional access control measures, designed to empower the patient rights and decisions on who is allowed to access his data (e.g., by containing consent information, representing a consent, or by containing cryptographic keys that are used to protect medical data.)

The fundamental idea of this card is that explicitly handing over the card to someone implies that the patient explicitly authorizes this data processing. In practice, this simple idea has some weaknesses when applied beyond the scope of a single software application/system because it is "all or nothing" and requires additional measures in order to be aligned with the purpose of use and compliance.

In cases where the card physically contains the security objects, which are required to perform data access, at least the inverse of the use case holds: Someone who has no access to the patient's card is definitely not authorized to access any of this patient's data.

4.1.4 Delegation of Access Rights

There are two major scenarios requiring delegation:

- Assigning **guardians** for incapable persons (either implicitly or explicitly named),
- Assigning **medical staff** to act on behalf of other people/roles (if this happens on a regular base, the "need-to-know" principle should be considered in the design of the respective policies instead of a per-case explicit delegation of rights)

These must be distinguished from "regular" assignments of access rights. Delegation is assuming some or all rights and responsibilities of another person. It is governed by law or by specific guidelines specifying which rights are delegated. In contrast to this, a regular assignment of rights to someone else is expressed by a consent.

The easiest way to implement a delegation relationship is through an attribute that is considered by a policy or by the policy decision algorithm. PEP may treat the guardian or staffer as if they were the other person for the purpose of access control decisions. The subject domain issues an authenticated subject information assertion for the subject (including a claim on the take over of someone else's rights) and an authentic subject information assertion about the delegator.

Subject replacement within the subject domain is a tempting but inappropriate solution, because delegation is taking over rights and not taking over identities.

4.2 Co-Existence and Integration of Policy Concerns

Multiple policy concerns may lead to multiple policies that have to be enforced during the flow of control - which leads from a request issued within the context domain to a resource access within the resource domain. In the ideal case this interplay of policies can be synchronized along the flow of control, while in the worst case it may cause deadlocks because of conflicting policies. In this section both of these aspects are discussed.

4.2.1 Policy Layering

Policies are defined for a certain level of detail. In practice, restrictions are initially set in rather abstract nature. By following the divide and conquer principle, policies of differing detail-level are established protecting different granularities of resources. To meet all requirements specified in such layered policies, various combination algorithms (e.g., permit-overrides) might be used.

In the simplest case, access policies and behavior policies can be sequentialized (cp. Figure 10):

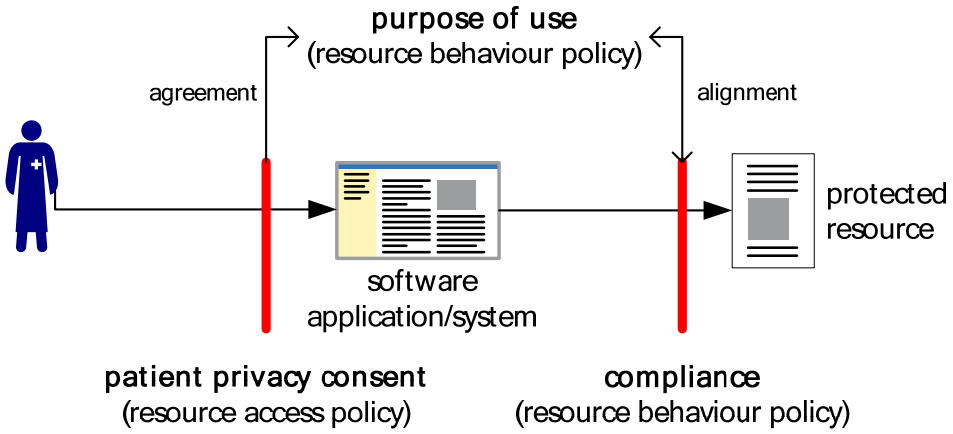


Figure 10: Sequentializing of Policies

In this case, the patient consents to a list of individuals who might access his medical data (protected resource) for a certain purpose of use by means of a software application/system (e.g., an EHR or a medication record). The software application/system handles all requests of authorized subjects further to the resource managing systems. These forwarded requests are intercepted (by a PEP) and their legitimacy is decided with respect to a compliance-driven resource behavior policy.

Reality - usually - is not that simple. For instance, just considering that a patient might grant access not to individuals but solely to organizations increases the complexity of the scenario. The compliance-driven resource behavior policy controls what individual/role is allowed to instantiate the organization's permission (e.g., is an oncologist at hospital A, allowed to open a cardiologic EHR for which the patient has declared hospital A as an authorized user).

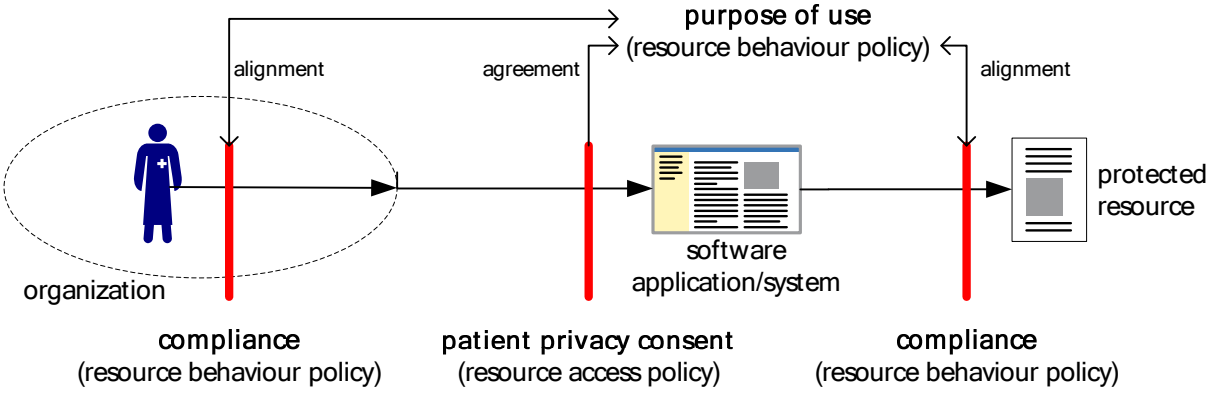


Figure 11: Compliance in a Distributed Scenario

It should be noted that in a distributed scenario (cross-enterprise) the compliance at the user side and at the resource side will be different. That might lead to scenarios where the software application/system is not able to realize the "need-to-know" relationship between the user and the resource (cp. Figure 11). This is illustrated in the next section. Further opportunities for combining policies are discussed in chapter 5 and appendix A.

4.2.2 Policy Conflicts

Traditionally, multiple policies need to be considered within an access control scenario. Especially if there is neither a dedicated sequential order of policy evaluation nor a prioritization of policies (see discussion on policy override in section 4.3.1), conflicts might occur.

A typical conflict is caused by patient consents, where specific individuals are excluded from resource access. However, those individuals may well have roles assigned within an organization that require access to exactly this resource for reasons of governance and compliance (e.g., quality assurance). In practice, this instance is quite likely to occur: the patient specifically prohibits Dr. Smith from accessing his data; however, Dr. Smith in his assigned role as surgeon would gain access.

In general two constellations might occur from policy conflicts:

- The legitimacy of an access request cannot be decided, because different policies apply and lead to conflicting/blocking decisions
- The end-to-end "need-to-know" relationship between an individual user and a protected resource is disrupted, because the policy that ensures this relationship is overridden by a permitting policy

A policy conflict is therefore always a conflict among concerns and can usually only be solved on the abstract level of the conflicting concerns:

- *Conflicts between patient privacy consent and purpose of use:* As stated in Section 4.1 a consent should always include the agreement for a defined purpose of use. On the other hand the purpose of use must define which configurations the patient privacy consent might impose on the derivation of a behavioral policy. Therefore any conflict between these two concerns is caused by a poor design and/or definition of the purpose of use.
- *Conflicts between purpose of use and compliance:* The purpose of use of the software application/system must be compliant to the organization's governance. Organizations should only agree to the operation and/or provisioning of software applications/systems after they made sure that these are not conflicting with compliance. For instance, if the purpose of use of a shared EHR application is not compliant with the rules of governance of a certain hospital, this hospital must not use this application.
- *Conflicts between patient privacy consent and compliance/organization of work:* These conflicts usually arise when a patient prohibits access to certain segments of his data to persons/roles that are in conflict with the internal organization of labor. For example, a patient requests cardiologic treatment by a hospital and the only cardiologist available is the person the patient did not allow to access his data. Conflicts of this kind can only be resolved by either modifying the consent or the organization of labor, e.g., by not assigning a certain person to this patient's treatment. There is no way to solve this conflict using technical means or special access control rules; a solution must always be approached on the organizational level and in conjunction with the patient.
- *Conflicts between multiple privacy consents:* Patients might have given multiple consents with multiple organizations that can conflict, if data is shared among these organizations. To prevent this kind of conflict, privacy consents should always be bound to a clear purpose of

use. This allows for the maintenance of a privacy consent history. This in turn is a precondition for applying technological means to retrieve the recent policy for a certain purpose of use.

- *Conflicts between multiple compliances in cross-enterprise scenarios:* The organization and division of labor differs from organization to organization. This leads to different compliance requirements. Therefore situations might arise in cross-enterprise data exchange, where a certain job role needs to know a medical document kept by another enterprise but is not able to access it because the resource side compliance does not grant access to this data for this role. Other than synchronizing compliance among cooperating enterprises (which is usually unrealistic), the best way to prevent these conflicts in advance is a clear definition of the purpose of use of cross-enterprise data exchange. By aligning this definition with the compliance of the cooperating enterprises, potential for conflicts may be discovered and resolved. This can be reached by realigning either the purpose of use or the compliance of the participating organizations.
- *Conflicts between multiple purposes of use:* Especially in health information networks that follow the paradigm of SOA, an access to a remote resource might be mediated through a hierarchy of nested services which all define their own purpose of use. In this case, only the permissions within the intersection of the called services' purpose of use will be preserved, which easily leads to a violation of the end-to-end realization of the need-to-know principle. This conflict can only be prevented by a proper design of nested services which takes the implications on their joint semantics into consideration.

Recommendation: Make sure that privacy consents always include an agreement of the patient to a clear purpose of use and the internal processes that are used to process his data. Make sure that especially software applications/systems shared among enterprises comply with all parties' rules of governance and that software application/system specific roles match with organizational roles. A chain of service calls is a service (i.e., service choreography). Make sure that the semantics of the top-level service is preserved throughout the whole chain. Always solve conflicts between consents and organization of labor on an administrative level.

4.2.3 Patient Safety

Patient safety is an important issue that has to be considered because of the additional points-of-failure and restrictions on flexible role assignment that might be caused by an access control solution.

A typical scenario that affects patient safety arises from emergency situations. It cannot be assumed that personnel that are authorized to access the patient's medical data is always close to the point of emergency care. In an emergency case, any available medical staff member must be enabled to access the patient's data in order to provide immediate help. This situation may be solved by deactivating resource access policies and using behavior policies that are activated and applied in an emergency.

- A medical organization could define a specific emergency software application/system (including the definition of roles, permissions, and obligations) that is focused on preserving the patient's safety in case of an emergency. Access to this application is open to all medical

staff members and the patient (implicitly or explicitly) agrees on the use of the application's purpose of use when he signs the treatment contract

- Emergency data access is covered by the organization's rules of governance and subject to IT compliance (e.g., by defining a break-glass policy).

Another scenario where patient safety matters, is the unavailability of technology that is required to manage, decide, or enforce policies. This scenario might occur in a scheduled manner (e.g., setting up a temporary hospital to serve people after a natural disaster) or unscheduled (e.g., because of a power failure).

Scheduled scenarios with limited IT infrastructure, restrictions on power consumption, and solely implied or presumed consents can be handled in a regular manner by defining appropriate purposes of use and considering the lack of specific patient consents as part of IT compliance.

4.3 Binding of Policies and Attributes

A policy statement "All cardiologists of hospital A are allowed to access the medical data of John Doe" contains a subject role attribute (cardiologist), a subject organization attribute (hospital A), an operation attribute (access), a resource defining attribute (medical data), and a patient identifying attribute (John Doe).

Each attribute is defined by:

- An **attribute stub**, which is used at policy design time and determines the semantic and syntactical encoding of an attribute. E.g., the attribute stub "patient identifier" could be defined as a unique identification number of the patient that is provided as an HL7 instance identifier encoding of the serial number of the patients electronic health card.
- An **attribute value**, which instantiates the attribute stub at runtime by providing a concrete value that matches the specification of the respective attribute stub.

The relationship among attribute stubs and attribute values is 1:many; each attribute stub might be instantiated by many different values, but each attribute value at runtime is bound to a defined attribute stub.

4.3.1 Policy Activation and Policy Decision

A strong interdependency between policies and attributes exists. Without the existence of appropriate attributes, the application of rules contained within policies is impossible. Attribute values are evaluated for both activating and deciding policies. Which attributes stubs are needed for either case depends on the policy and therefore is highly dependent on the policy concerns.

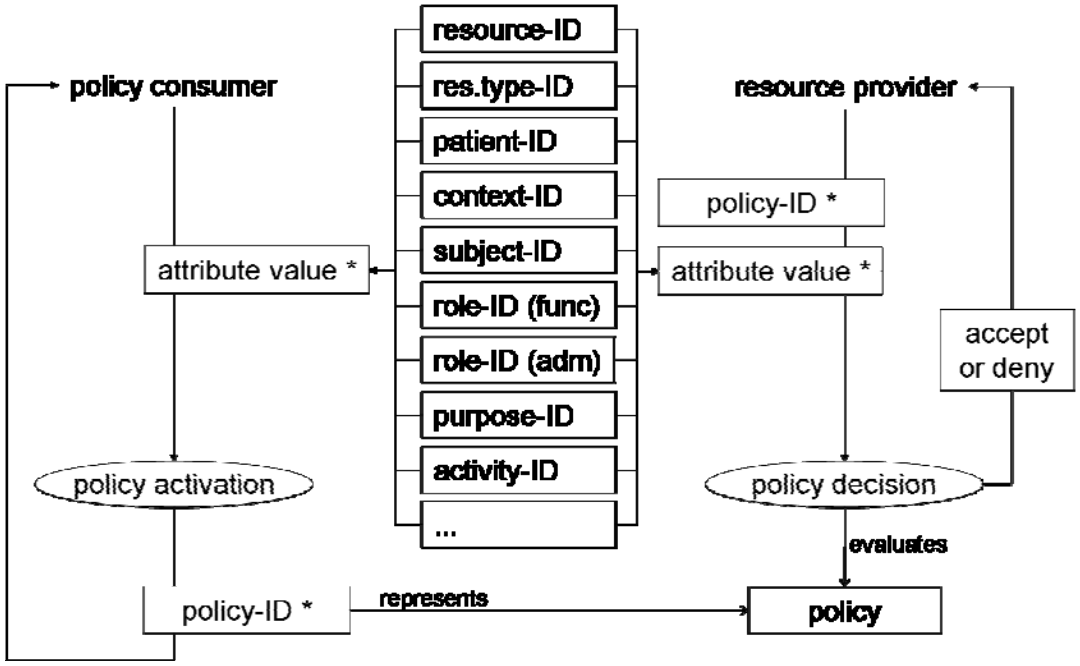


Figure 12: Relevant Attributes for Policy Activation and Policy Decision

Figure 12 depicts the demand of attributes needed for policies in two processes. It depends on the purpose of use which attributes are needed for policy activation and rendering a policy decision. Taking the above example of a resource access policy ("All cardiologists of hospital A are allowed to access the medical data of John Doe"), the attribute values "access", "medical data", and/or "John Doe" have to be fully matched with the request message in order to decide whether this policy statement has to be evaluated.

Only if the desired request is an access to the medical data of John Doe, the policy statement will be selected and "activated". The attribute values "cardiologist" and "hospital A" have to match with the subject's role and organization attributes in order to decide on the legitimacy of the request (cp. Figure 13).

Usually the syntax of a policy language is designed in a way that the part of the "sentence", which is relevant for policy activation, is clearly separated from those statements that have to be considered for policy decision. This enables a separation of the respective attribute value flows required for the policy activation and the policy decision. Therefore, the separation increases the design opportunities of the general flow of control among the ACS of the different domains (see annex B for an example).

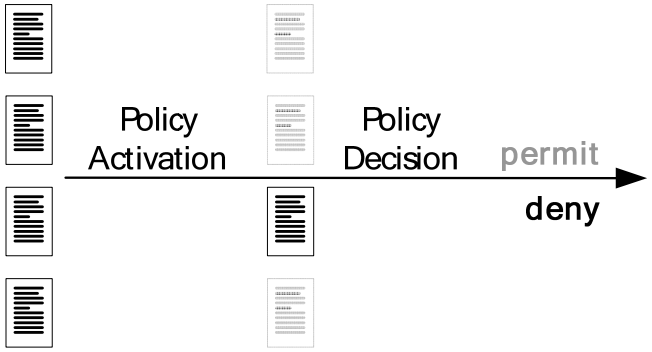


Figure 13: Process of Policy Activation and Policy Decision

Policy activation subsumes the concept of policy override. E.g., in an emergency scenario a dedicated emergency access policy might overrule all other policies including the ones derived from the patient's privacy consent. Meta-policies like this must be implemented by the policy activation actor. This again requires that even all attributes required for deciding on this meta-policy must be provided for policy activation, too.

Recommendation: Clearly work out which attributes are required for the activation (selection) of a policy and for the decision on a policy. Consider attributes that are needed to decide on the implicitly defined meta-policy (e.g., override of all other policies by an emergency policy in case of an emergency access). Define the respective attribute stubs in a formal way; e.g., by mapping them onto data types or message fragments. Reason about the security status of these attributes by analyzing possible threats that might arise from corrupted attribute values. Define conditions that might require specific means to preserve the confidentiality, integrity, or authenticity of attribute values (e.g., transferring authenticated subject identifiers and role assignments over a public network).

4.3.2 Attribute Sources

At the point in place and time, where a policy is activated and decided the required attribute values for the respective attribute stubs have to be available. With respect to the generic model of policy based access control (see section 3.1.3) this can either be realized by the requestor, who includes them with the request message or by the processing party who retrieves them on demand from a policy information point.

In either case it must be known through the respective attribute stubs, which attribute values are required and which source can be used to retrieve them (Figure 14).

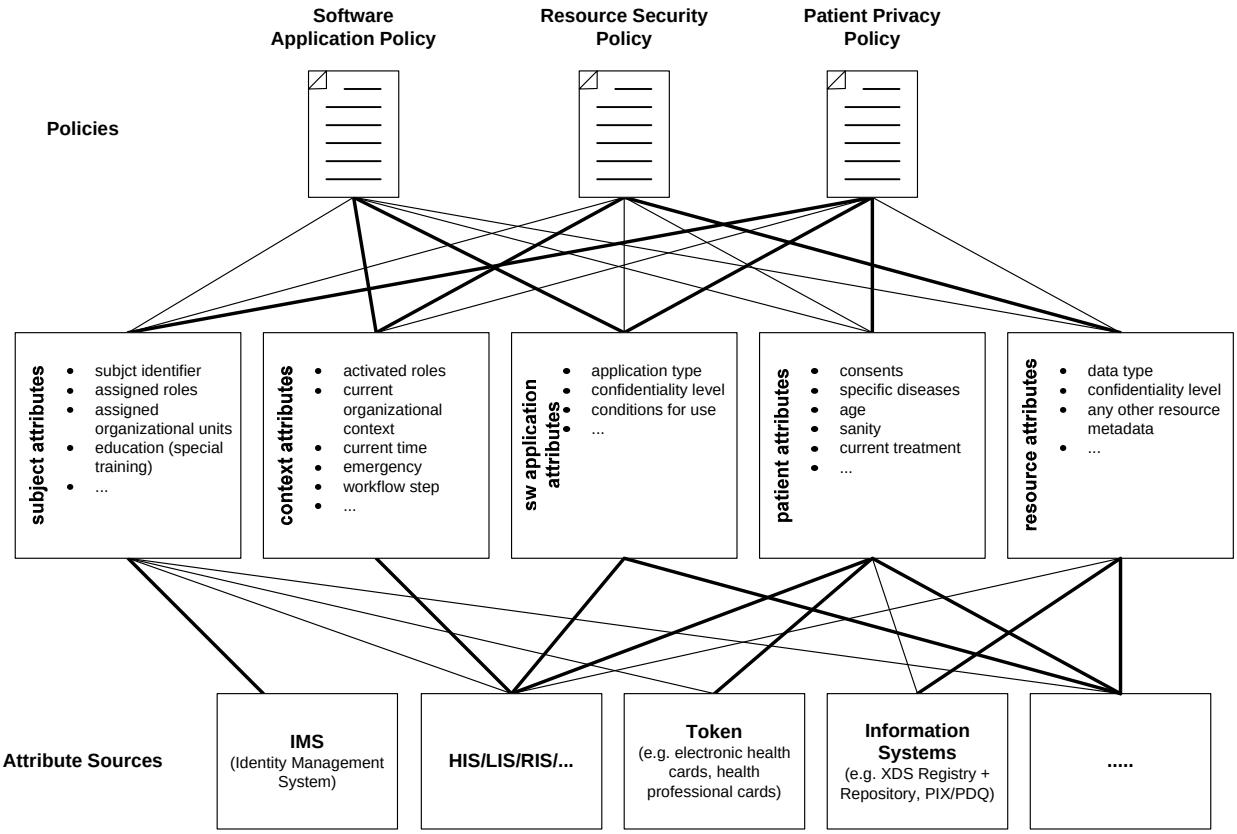


Figure 14: Attributes and Attribute Sources

From a theoretical point of view, each policy should be able to reference the full set of attributes and each attribute value that may be presented by the different sources. In practice, the concerns of the policy limit the types of reasonable attribute stubs and the type of an attribute stub limits the set of reasonable attribute value sources:

- **Subject attributes** provide additional information on the user that tries to access a resource. Subject attributes can be used to define and evaluate rules that refer to certain characteristics of the accessing subject. The most prominent example for a subject attributes is a role assignment (e.g., "*Dr. John Doe is a cardiologist.*") The main source for subject attributes are standalone identity management systems or identity management components that are integrated into the HIS or similar systems.
- **Resource attributes** provide information on the requested resource and are widely used in resource security policies. A prominent example might be the confidentiality level of an accessed information object or the information type or class (e.g., "*A metadata entry for a medical information object shows that the requested resource contains sensitive medical information.*"). Resource attributes can often be derived from resource metadata (e.g., contained in registries).
- **Context attributes** refer to activities, purposes, or the context of an intended resource access. Prominent examples are the activated roles that are assigned to subjects, or certain process or workflow steps (e.g., "*Medical information is requested within an emergency*

context."). The main attribute source can be identified in the request message itself and in systems that control the information workflow, e.g., HIS or LIS.

- **Software Application/System attributes** refers to characteristics of a software application/system. Examples of application attributes are confidentiality levels and limitations in the purpose of use that might lead to obligations on certain operations (e.g., "Every access that is mediated through applications with full access to a patient's data has to be logged."). Application attributes are, in many cases, hard-coded within the application and mapped onto other attributes (e.g., resource and context) during the mediation of an access operation.
- **Patient attributes** refer to the patient, his characteristics, and wishes. Prominent examples include attributes concerning the sanity or age of the patient and attributes expressing his consents (e.g., "*The patient documented his consent to use a certain healthcare application on his electronic health card.*"). Sources for patient attributes are, among others, patient management systems and electronic health cards.

Recommendation: For each attribute stub that is required for either policy activation or policy decision, identify the respective attribute value sources. Identify which further attributes might be required to retrieve a certain attribute value from an attribute value source (e.g., retrieving a subject role will require providing a unique subject identifier). Define the respective attribute stubs and align them with previously defined stubs in order to limit the number of representations of attribute values (e.g., things are much easier, if a subject identifier needed for retrieving a role attribute is defined the same way as a subject identifier that is needed for deciding on a policy that is derived from a patient privacy consent). Verify, whether the analyzed level of security for attributes values can be provided by the attribute source or if it has to be provided by specific means within an ACS. Elaborate restrictions that hold for retrieving attribute values from an attribute source. E.g., an attribute service that provides role information might only be locally accessible or require a prior login which would place restrictions on the flow of control and on the deployment of functionality among the ACS.

4.3.3 Policy Profiles

Standardized languages for encoding machine-readable policies usually offer a wide variety of opportunities for the placement of attributes and their encoding. Especially if multiple instances of policies (e.g., reflecting the consents of thousands of patients) have to be designed, created, and managed, it is highly advisable to consider a set of certain risks. Policy authors may be tempted to use attributes that have not been fully considered during the design of the access control solution. Subsequently, those attributes may not be bound to attribute value sources or may not be evaluated during policy activation/decision because of an encoding that does not match the formats used throughout the rest of the system.

A policy profile can be used to restrict the expressiveness of a policy language to the attribute stubs that have been defined in advance. Such profiles should define a minimum of:

- Which attribute stubs can be used for defining the policy (including the restrictions of the encoding formats as defined by the attribute stubs)

- Which attributes are mandatory (e.g., because they are required for policy activation) and which are optional
- How should the various policy elements (conditions, rules, etc.) be placed and nested in order to allow a common – and therefore efficient – processing of a policy

Recommendation: Define profiles for each policy. Use tools for policy encoding that allow for the definition and enforcement of policy profiles. Provide means within (or close to) the policy administration point to ensure that each registered policy matches the requirements of the respective policy profile.

An example is given in Appendix B.

5 Use Case and Analysis

Among the most prominent decisive factors for the concrete design of the ACS interaction and its flow of attributes and polices are:

- Security assumptions about the nodes involved; e.g., what is the current level of environmental protection/security of the desktop systems that are used by medical staff members?
- Processing power at the nodes involved; e.g., is there a requirement to use thin clients?
- Accessibility of attribute and policy sources; e.g., are certain subject or patient attributes maintained on physical carrier mediums (e.g., smart cards), that may not be accessed by every system and at every time?
- Availability requirements; e.g., does a shortfall of the security subsystem hinder physicians to do their work and which alternative processes are in place?
- Performance requirements; e.g., how time-critical are the safeguarded business processes?
- Consequences of policy misinterpretation; e.g., what damages might be caused if the intention of a security policy is not totally reflected by its implementation or not enforced because of a system malfunction?

The following sections demonstrate that there are multiple opportunities and various approaches to the deployment of policies and attributes and on the flow of control among distributed access control systems. However, regardless of their actual implementation and problem solving approach, all provide the same functionality: *the integration and enforcement of the policies that apply for the given access control scenario*.

The only difference among these options is how they cope with certain non-functional requirements and environmental conditions. It is impossible to identify a "one-fits-all-solution" that perfectly supports all requirements. However, for most scenarios there is a concrete solution that fits the particular requirements and addresses the individual demands of the given business processes in full respect to the actual organizational environment.

5.1 Sample Use Case

Throughout this chapter, an example scenario is used in order to demonstrate the effects that varying flows of attributes and policies have on certain non-functional requirements and environmental conditions.

***Storyboard**: In a metropolitan area several hospitals set up a network for the exchange of medical patient data. It has been observed that many patients visit different hospitals for different purposes and diseases. Reasons for this are the high density of hospitals, the medical specialization of many of these hospitals, and the high degree of transparency with respect to treatment-specific quality parameters. During treatment, physicians often become aware of previous treatment at other hospitals and the existence of diagnostic data that might be useful to consider in order to evaluate the severity of ongoing medical conditions and to verify a suspected diagnosis.*

To support this situation a dedicated healthcare application system is designed on top of the hospital network that enables physicians to easily access historical data from other hospitals that might be relevant (e.g., lab data, radiologic data). An access to this data is only permitted after the patient has consented:

1. *What kind of data might be processed,*
2. *Which organizations might have processed this data,*
3. *Which roles are authorized to process this data, and*
4. *For what purpose the data might be processed.*

The use of the application system and its purpose of use requires consents to be given in advance; either after a stay at a hospital (for future encounters) or while checking-in for ambulant or stationary treatment (for the current encounter).

For the sake of clarity, the focus of the following discussion will be on the enforcement of the patient privacy consents. It is assumed that all participating hospitals agree on a role model and will allow cross-hospital data access for well defined roles. This requires full enforcement that those inter-hospital roles are the ones which the patient is explicitly naming in his consent.

5.2 Methodology for ACS Interaction Design

In order to elaborate the flow of attributes and policies and the configuration of the ACS building blocks, a seven-step methodology is used:

1. Design a policy profile and identify the attributes needed to instantiate the profile (see Section 4 and Section 5.3).
2. Collect and prioritize any immediate and collateral requirements (see Section 5.6).
3. Identify the attribute sources and domains where these attribute can be made available (see Section 5.4).
4. Define the default flow control among the client-side and resource-side ACS (see Section 5.5).
5. Optimize the process flow of the ACS with respect to the prioritized requirements (see Sections 5.6 and 5.8).
6. Deploy the (logical) domains onto physical nodes (see Sections 5.7 and Appendix A).
7. Map the ACS building blocks and the messages among them onto actors and transactions (see Chapter 6 and Appendix C).

5.3 Policy Profile and Attribute Stubs

A consent template is to be defined that can be mapped onto a corresponding policy profile. For the sample scenario this template will look like:

*I hereby authorize **[roles]** at **[organizations]** to use the "Historical Database" Application in order to access all **[Patient] [kind-of-data]** for the purpose of **[purpose]**.*

A valid instance might be:

*I hereby authorize **physicians** at **Clinic A** to use the "Historical Database" Application in order to access all **my lab data** for the purpose of **medical treatment**.*

In this sample scenario the purpose of use is included twice: It is directly mentioned (*medical treatment*) and implicitly derived from the term *Historical Database Application*. This is due to the fact, that the patient might have the possibility to configure resource behavior policies (cp. Figure 7 configuration). The phrase *purpose of medical treatment* represents such a configuration scenario. Another configuration might be the phrase *purpose of medical research*. By contrast, the derived purpose reflects the mapping of a concrete intended use to a software application/system.

Following the process described in Chapter 4, one early step in the design of an access control system is to identify which attributes are required for policy decision and policy activation. The result of this is a set of attribute stubs that have to be considered in the design of the access control system in terms of how to retrieve and arrange them for policy activation and decision.

For the sample scenario, there is one consent per software application/system and patient. Therefore, the attributes required for policy activation are identifiers for these two factors. Policy decision is based on the concrete attribute values that instantiate the policy template's variable places for the activated consent. Figure 15 shows the attribute stubs together with their respective source classifications and sample instance values (see Section 4.3.2).

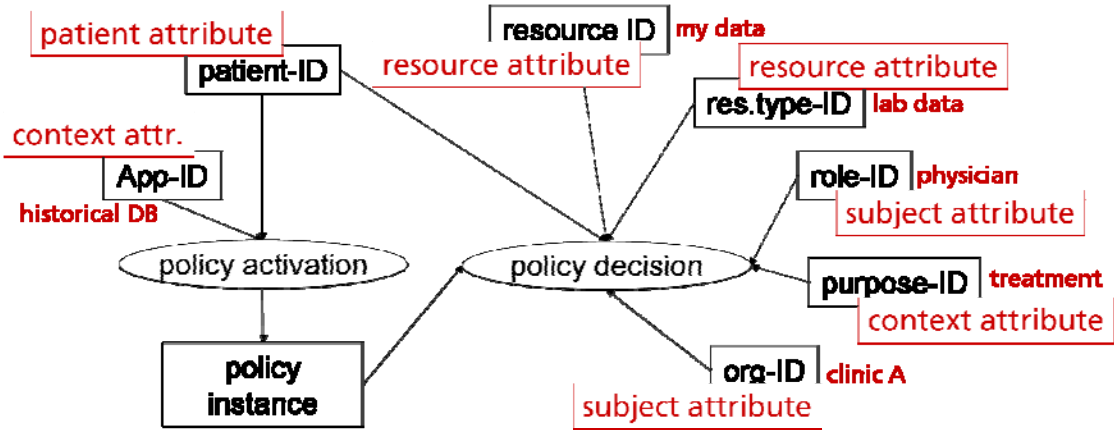


Figure 15: Attributes for Policy Profiling

5.4 Attribute-Domain Mapping

In Section 3.3.2 the notion of an access control model was sketched that consists of three core domains, which are common for any system:

- A protected resource,
- A subject wanting to access this resource,
- And a current context from which the access request is issued.

In Chapter 4, additional sources for attributes and policies were identified. One clearly is the patient who is the owner of his medical data. Other sources are applications acting as semantic frameworks for resource operations. These can be integrated in the model by deriving two additional domains:

- Patient domain, where attributes about the patient and the patients consent together with derived policies are located
- Application domain[6], where attributes about the application are located. These attributes represent the application's purpose of use as well as related policies. In most real world scenarios the application domain will be rather implicit because the semantics and the implied policies of an application are rarely expressed explicitly (see e.g., the example of the eCR in annex A where the application defines certain specific roles and their behavior but does not express this by something machine-readable).

In the case of an XDS Affinity Domain the application domain might act as a representative for the XDS Affinity Domain by (virtually) managing its agreed rules of governance. For instance, if all members of the XDS Affinity Domain agreed upon certain access restriction for specific roles or demand for specific authentication means, the respective polices would be logically located within the application domain. Even though in most cases these policies are rather hard coded with the resource managing systems or integrated with the resource managing side's internal security policies.

Further domains may be defined whenever required in order to enable a potential matching for even more complex scenarios (e.g., multiple resource domains and hierarchies of application domains to reflect scenarios where multiple XDS Affinity Domains are involved).

Figure 16 shows the fundamental domain model as it can be used for designing access control solutions for healthcare scenarios. The main trust brokerage mechanism among domains is the exchange of security tokens, which have been issued by trustworthy services. Each domain holds its assigned attributes and policies. Any required policies and consents must be explicitly activated (selected) before use. Domains, in which access restrictions may be enforced, usually feature policy enforcement and decision points (a deeper insight into the building blocks of the various domains is given in Section 5.6).

[6] In this context, the term application should be understood as another abstraction of a clearly defined purpose of use. This is due to the fact, that the purpose of use can be reflected and implemented by dedicated software applications/systems.

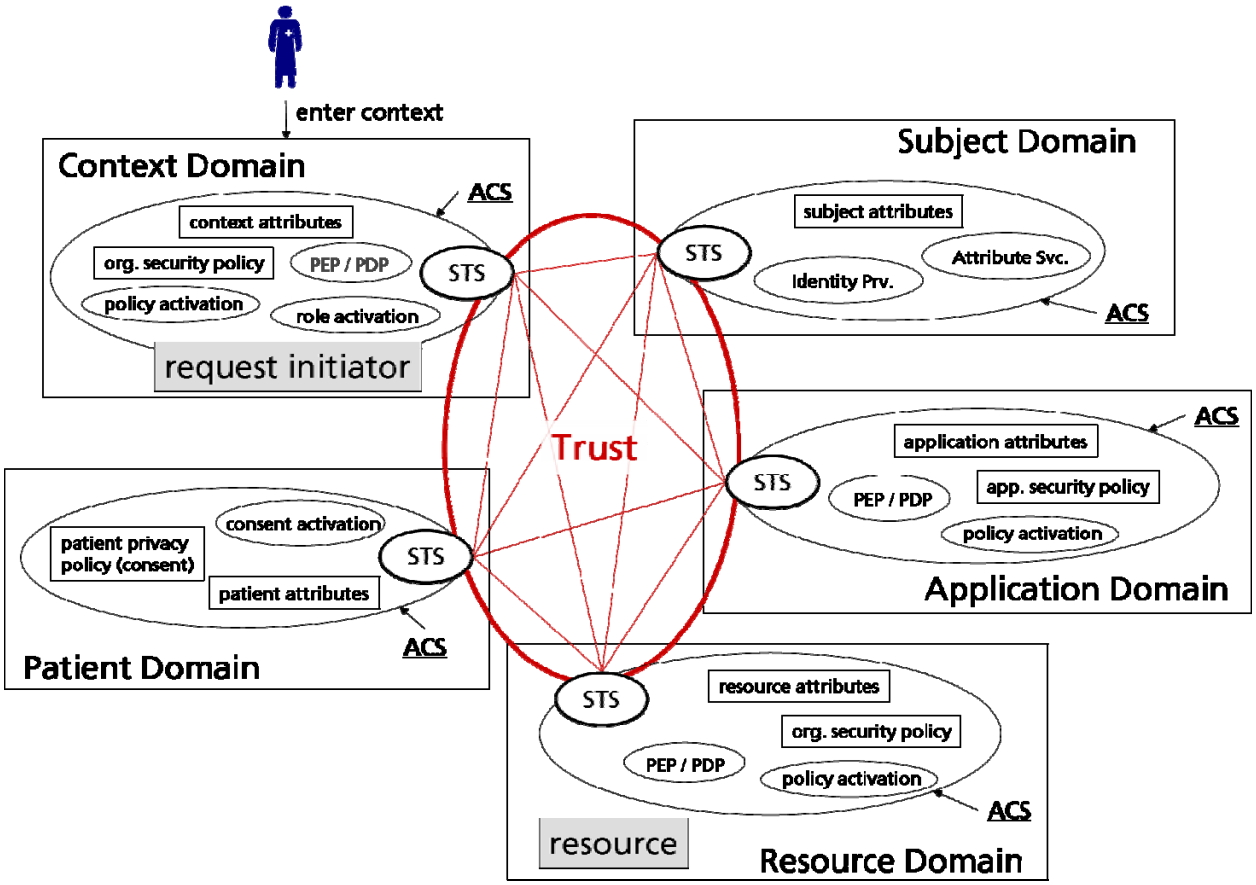


Figure 16: Attribute Domain Mapping

The next step in designing an access control system is the assignment of the previously identified attribute stubs and policies to the domains. The rule for this assignment is that each attribute stub and policy is assigned to the domain where its value is either maintained by a respective service or where it is implicitly known. For instance, the attribute "application identifier" is assigned to both the application domain and the context domain. In the context domain it is implicitly known, because the user makes use of this software application/system in order to access a protected resource.

Furthermore, it has to be determined which attributes' values are explicitly expected to be known, when the subject (physician, etc.) enters the context domain. In most cases this will at least include the subject identifier (assuming the consumer knows who he is) and a patient identifier (assuming the consumer knows, which patient's data he wants to access).

Figure 17 shows this assignment of attributes and policies to domains for the sample use case.

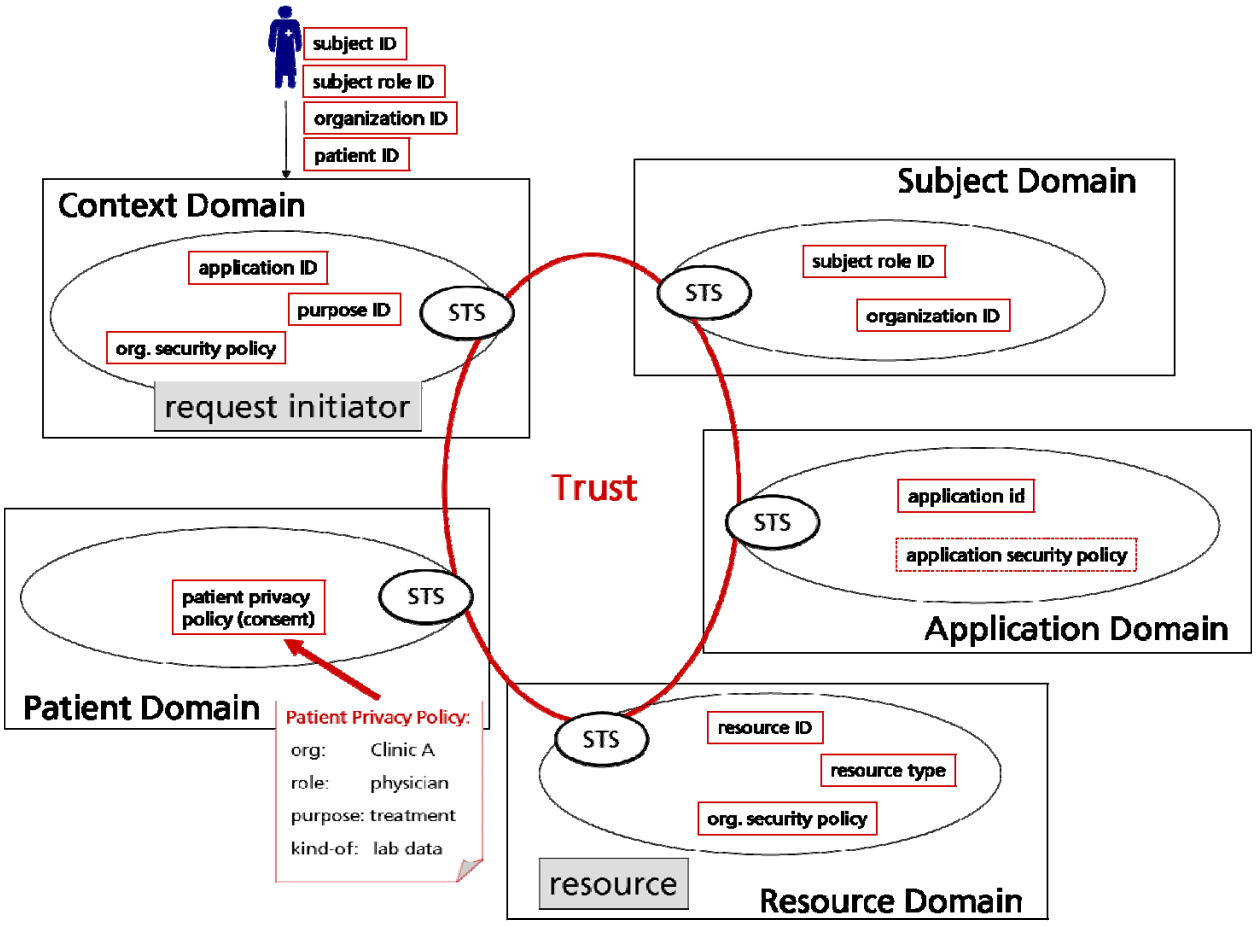


Figure 17: Assignment of Attributes and Policies

5.5 Default Process Flow

Security functionality should – whenever possible – be positioned on a technical layer below the business services. One objective of that is, not to burden or hinder business functions and business process flows with security functionality. Therefore, any potential future adjustments of the security functionality – for instance due to changing legal requirements and environmental conditions – may be performed without having to modify actors and transactions on the business level.

The starting point for the design of the security process and the flow of security objects is always the analysis of processes at the business level. For the example scenario, the business process flow is rather simple, as it is just a consumer (subject) who wants to access a protected resource through the means of a defined application (Figure 18).

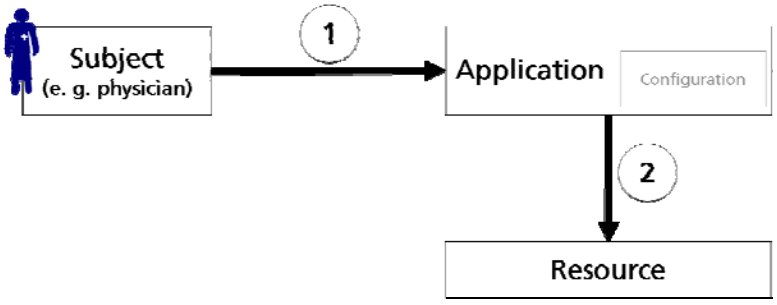


Figure 18: Process Flow at Business Level

The next step of the design of an access control system deals with the integration of access control into the business flow control. It covers three different aspects:

- The safeguarding of subject attributes (authenticity),
- The initial placement of policy enforcement, and
- The provision of required attributes values for policy activation and policy decision.

5.5.1 Authenticity of Subject Attributes

Authorization closely relies on authentication. For deciding on the legitimacy of a request, the ACS receives claims about the identity of the accessing subject (e.g., role and organization memberships). The whole security of authorization depends on the trustworthiness of these claims (e.g., if every user of a system can himself assign the role of an administrator, the access control system will grant every user the access rights of an administrator).

Therefore, all subject information that is needed for activating a policy and its subsequent decision must be assigned and vouched for by a trusted party. A common way to do so is to introduce an identity provider, which is a specific instance of a subject domain security token service.

The core functionality of an identity provider is to authenticate the claimed identity of a subject by verifying its provided credentials (e.g., a password, a signature, or a response to a challenge). By interlinking an identity provider with subject attribute services (e.g., a directory of staff members) and/or context-sensitive role activation, the issued identity claim can also contain trustworthy information on roles and organizational memberships.

In general, there are three approaches to authenticate subject information:

- Subject authentication and role activation is requested by the context domain whenever a user enters or activates this context,
- Subject authentication is implemented in a context-independent manner and authenticated subject information is provided to the context domain when it is initially entered (single sign-on), and
- Subject authentication and/or role activation is requested on demand whenever deciding on a policy.

The first two approaches are to be discussed below while on-demand role activation is covered in Section 6.5.

Authenticating the subject on entering a context is quite common. For instance, if the context corresponds to a radiology system, the user usually logs in to that software application/system before he can access any data.

The only difference between a local log-in with an software application/system and a subject authentication for additional processing by an ACS is the scope of trust: While a local log-in is only trusted within the secure scope of this application, an authenticated subject information issued by an identity provider can be trusted outside the context of the issuing application and even in case of trust brokerage (cross-enterprise scenario).

Authenticated subject information is usually encoded as a set of claims about a set of certain subject attributes, which are digitally signed by the issuing service. Cross-enterprise user assertions as defined by the IHE XUA integration profile are an example of a healthcare-specific encoding of authenticated subject information.

Figure 19 shows how this authentication patterns looks like for the sample scenario:

1. A user enters a context (e.g., a radiology system).
2. During context activation the subject's identifier in conjunction with authenticating credentials – and potentially additional information for role activation – is transmitted to the subject domain.
3. At the subject domain the user's credentials are verified and – upon success – a cross-enterprise user assertion is issued. This assertion states the successful authentication of the user and contains further attributes on roles and organizational memberships.

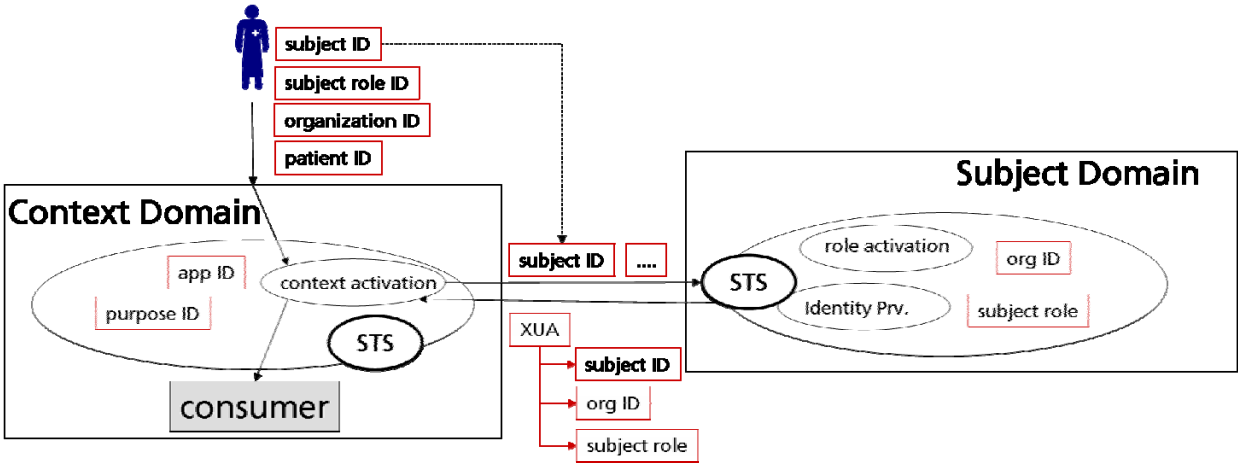


Figure 19: Authentication by Means of XUA Assertion

By performing subject authentication outside the boundaries of a specific context domain, a single sign-on among multiple contexts can be implemented. The advantage of this pattern is that the user has to provide his credentials only once. The drawback is that context-specific role activation cannot be performed within the subject domain and therefore has to be implemented by each context. For this reason many software applications/systems using a single sign-on operate on the full set of the user's potential roles without restricting it to the specific roles needed for the current purpose of data processing (which might result in a violation of the "least privilege" principle of secure design).

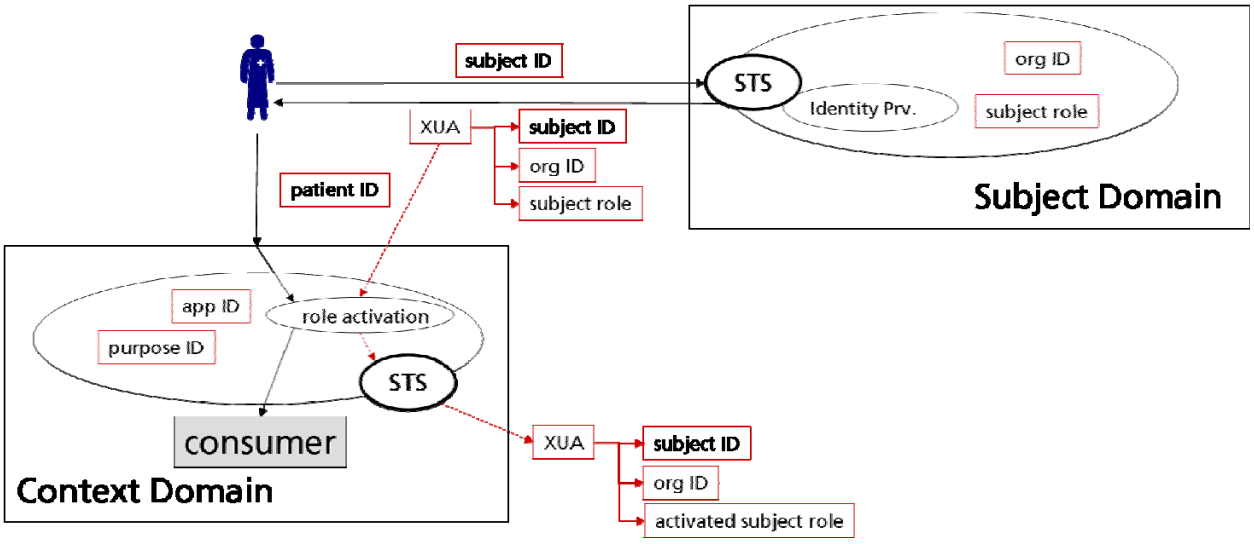


Figure 20: Authentication by Means of XUA Assertion with Single Sign-On

Figure 20 shows how the sample scenario might look like when single sign-on is used. In this deployment model, the role activation functionality is located within the context domain. Further options for deploying role activation (e.g., at the resource side PDP) are discussed in Appendix A.

5.5.2 Policy Enforcement and Policy Decision

Following the security design principle of "complete mediation", each request to a protected resource must be explicitly routed through a policy enforcement point. Since the number of possible paths to a resource increases with the distance from the resource (e.g., 50 business services use 10 different services that all are built upon a single system that maintains the resource), a rule of thumb is to place the policy enforcement as close to the protected resource as possible (cp. Figure 21). Exceptions of this rule are discussed in Section 5.7.

This results in PEP and PDP to be located within the resource domain, intercepting the access request message.

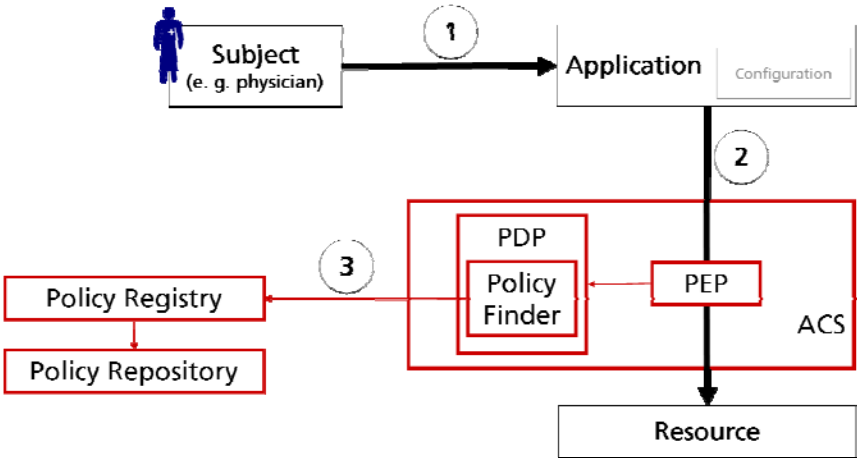


Figure 21: Resource-side Policy Enforcement

With the PEP and PDP positioned at the resource, the consumer of a protected resource must forward all information required for policy activation and decision to the resource provider's domain. It is then the responsibility of the PEP to extract this information from the intercepted message and to forward it to the PDP in a well-defined manner.

Figure 22 shows how the attributes defined for the sample scenario are exchanged among the different domains in order to activate and evaluate the appropriate patient privacy policy.

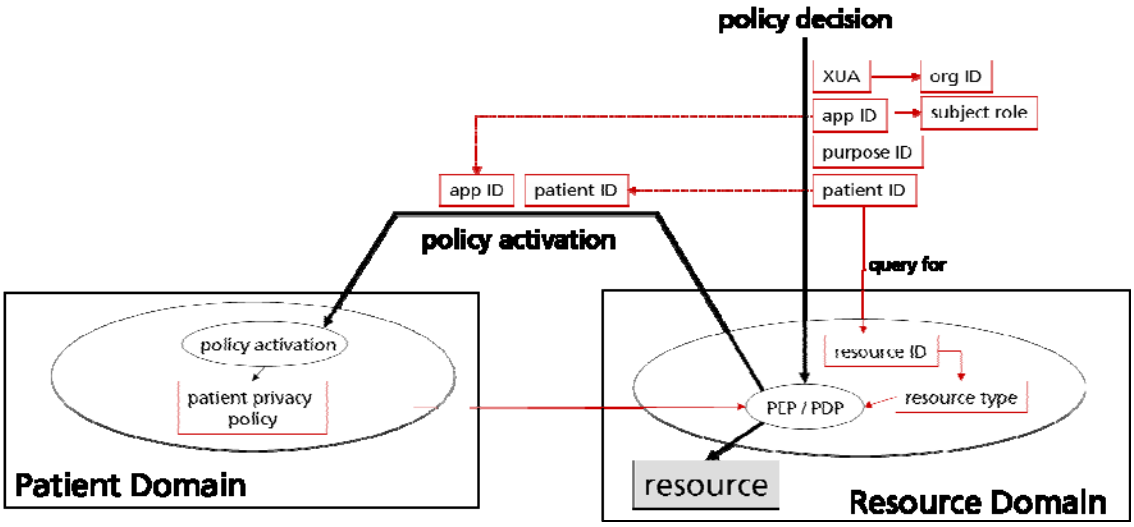


Figure 22: Attributes Exchange while Policy Activation & Policy Decision

The transmission of access control related parameters among business actors (e.g., resource consumer and resource provider) is done by piggybacking this information with the business transaction. Whether these additional arguments are visible to the business services depends on the communication protocols used.

5.5.3 Default Flow of Attributes and Policies

Figure 23 shows the default flow of attributes and polices for the sample scenario. Business level transactions are denoted with black arrows while access control related transactions are shown by red arrows.

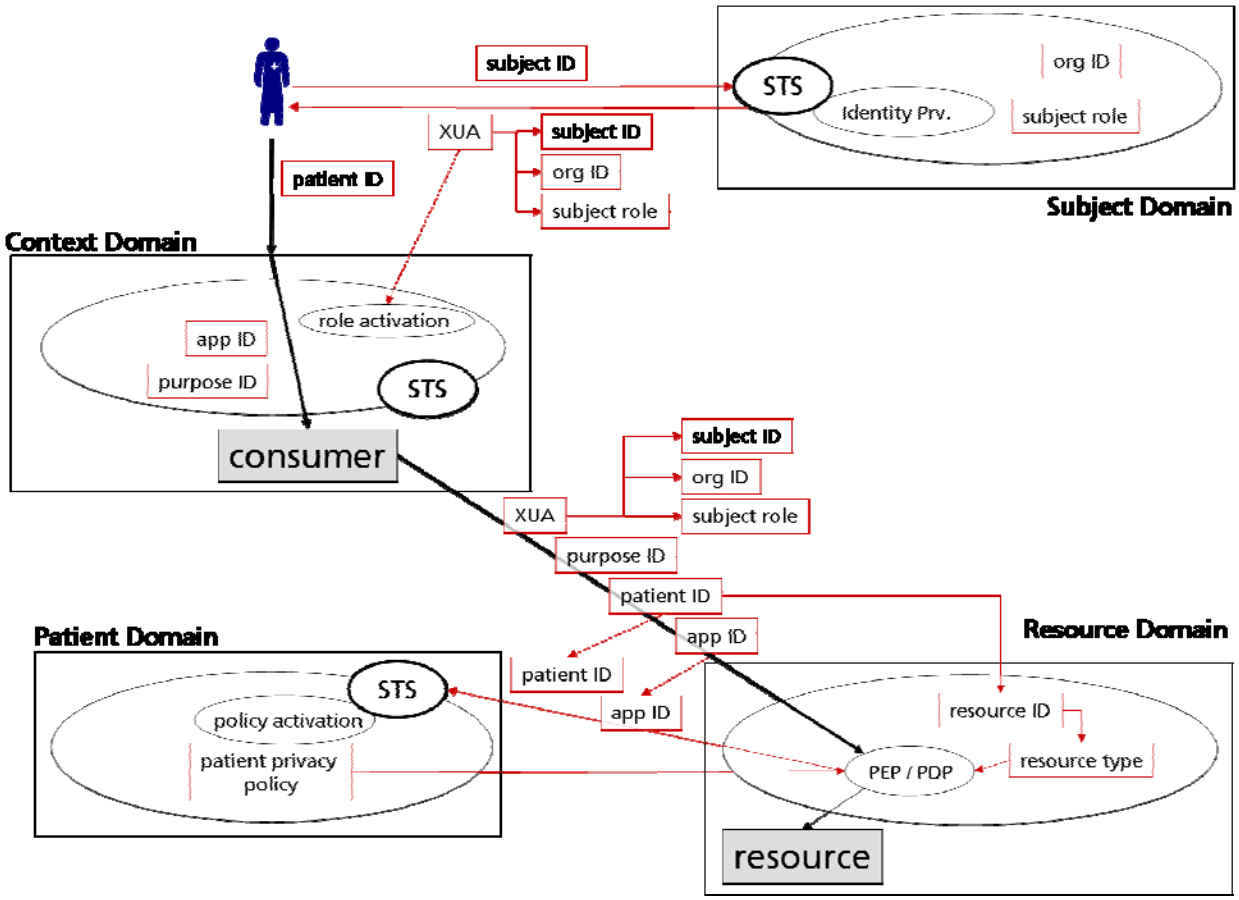


Figure 23: Default Flow of Attributes and Policies

5.6 Opportunities for Adjustments

The default security process flow as described in the previous sections fulfills all functional requirements on access control: It makes sure that only access requests to protected resources are performed that comply with the policies that hold for these resources.

It also recognizes the common principles of secure design, e.g., *complete mediation* by placing the PEP close to the resource and *least privilege* by context-specific activation of roles.

However, aspects that are not fully addressed in the proposed model and methodology are:

- Volume and number of transactions,
- Assumptions on the environmental security level,
- Functionality available at the consumer node,
- Single points of failure, and

- Potential emergency overrides.

Therefore, the next step of the methodology described in this white paper is the adaptation of the model to the given environmental setting.

Optimizing the flow of attributes and policies with respect to a concrete environment is done by shifting security related actors and responsibilities between domains. The four main variations that will be discussed in this white paper are:

- Which domain takes responsibility for obtaining the attributes needed for activating and deciding a policy,
- Which domain takes responsibility for obtaining the policy,
- Which domain takes responsibility for enforcing the policy, and
- Which domain takes responsibility for evaluating the policy.

Figure 24 shows how these variations address certain flows of control among the building blocks of the ACS at the different domains.

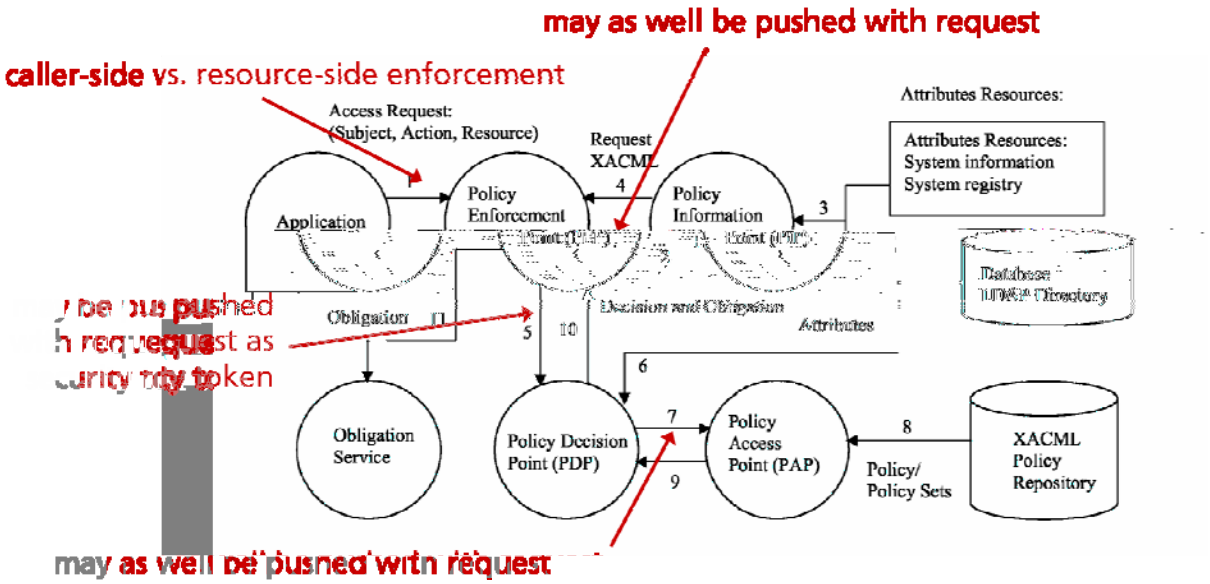


Figure 24: Flow of Control Variants

All of these variants can freely be combined, which potentially leads to a huge number of different attribute and policy flows that all implement the same access control functionality for the same business scenario. For instance, the default flow as sketched in the previous section implements resource side enforcement and decision by pulling policies and pushing attributes.

5.6.1 Policy Pull, Policy Push, Policy Cache

In the default model the resource side's *PDP pulls the policy* by sending a respective request message to the policy activation actor at the patient domain. An alternative pattern is the context domain pulling the policy and then *pushing the policy to the PDP* as a security token.

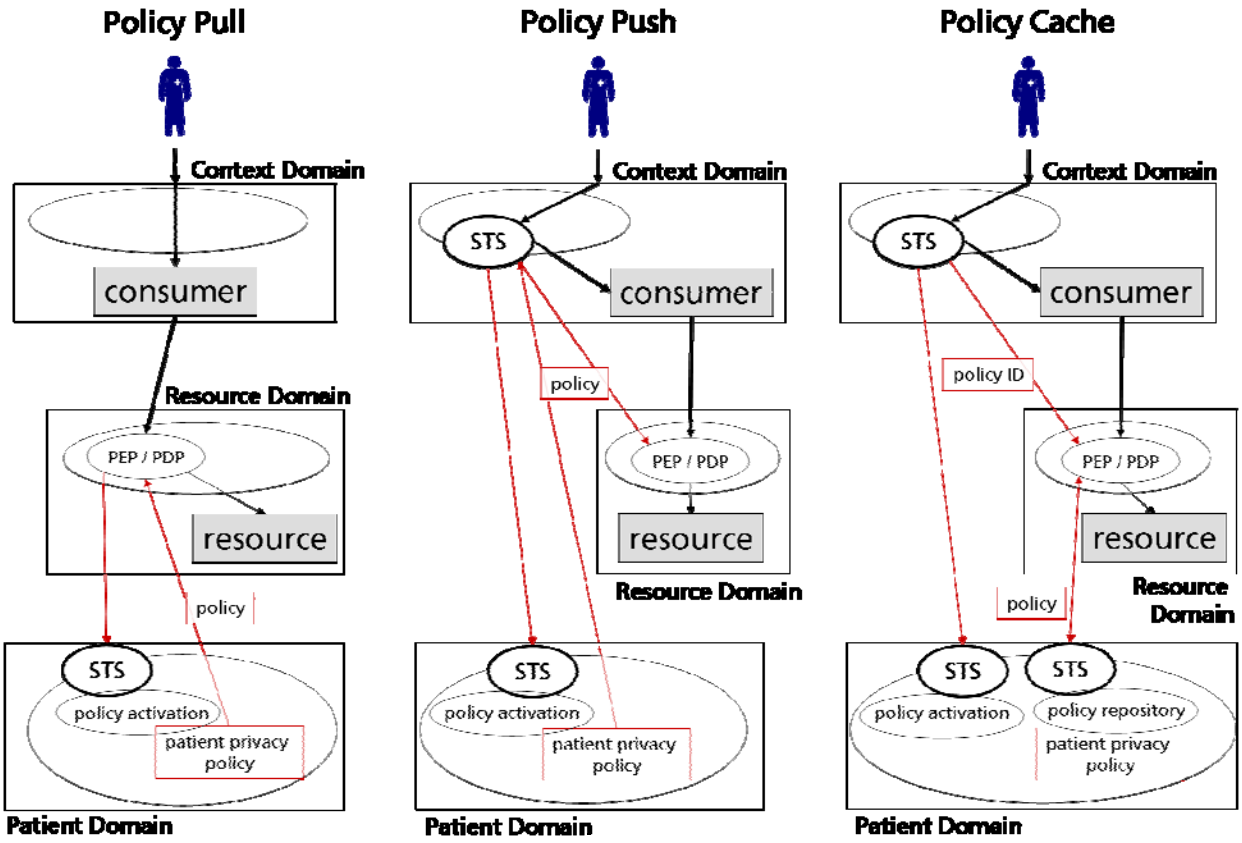


Figure 25: Policy Flow Patterns

Both approaches can be combined by separating the activation of a policy from its retrieval. Using this flow control, the context domain just queries for the identifier of the activated policy, then pushes this identifier to the PDP, which subsequently pulls the policy from a policy repository. This pattern enables the PDP to *cache policies* by their IDs.[7]

The following table lists the strengths and weaknesses of the three patterns.

Pattern	Strengths	Weaknesses
Policy Pull	Client does not have to care about policies PDP does not have to trust the client (if authenticity of subject information and policies is safeguarded by subject and patient domains)	Resource domain must be able to discover policy source and to establish trust with the patient domain's STS, which might be a problem in highly distributed scenarios
Policy Push	Client discovers policy, which is in most cases (patient-bound policies) more efficiently than policy discovery by resource side PDP	Policy must be transmitted on the security level of the business transaction which increases the size of this message (encoded policy plus signature)

[7] In environments with a middleware following the "service bus" paradigm (e. g. CORBA or ESB in SOA) the (semantically) lower-level "Cache Mediator" pattern can be used to implement the higher-level "policy cache" pattern by not caching policies at the PDP but instead mediating all requests for polices through an cache within the service bus. The "Cache Mediator" pattern must not be used in conjunction with the policy pull pattern because due to context parameters or information dynamically retrieved by the policy domain even the same input parameters might lead to another policy to be activated.

	Does not require the transmission of information on the current context and purpose of use to the resource side PDP Allows for client-side policy enforcement and/or decision	Additional means required to safeguard the binding of a policy to a certain subject and context (which is implied if PDP pulls the policy)
Policy Cache	Very efficient if a small set of patient-independent policies covers a large set of scenarios, which is usually the case for behavior policies	Additional means required to safeguard the binding of a policy-ID to a certain subject and context

5.6.2 Attribute Pull, Attribute Push

In the default flow control among the different ACS components, the context domain is responsible for the attribute discovery and retrieval. The client-side ACS component collects all attribute values that are required for policy activation/decision and pushes them to the resource domain.

Another approach is that the resource side PDP is responsible for obtaining the attribute values needed for activating and deciding a policy. This requires the availability of so called "policy information points" which can be queried in order to provide values of defined attribute stubs.

Both patterns can be mixed by deciding per-attribute which values are *pushed* to the PDP and which are *pulled* by the PDP. For deciding whether a certain attribute's value should be pushed or pulled, the following rules of thumb should be applied:

Recommendation: Attribute values should always be pushed to the PDP, except for cases where
- the retrieval of an attribute's value requires information that is only available at the resource domain
- the context domain is unable to either discover an attribute's source or to establish a trust relationship with the attribute's source

If no mapping of attributes onto policies has been done at design time, the context side ACS components may not "know" which attribute's values are required for policy decision. In this case the context side ACS component should provide the PDP with all attributes they can gather (e.g., authenticated subject information, context identifier, patient identifier) and rely on the PDP to be able to pull all other attribute's values on demand.

5.6.3 Client Side PEP/PDP, Resource Side PEP/PDP, Intermediary PEP/PDP

Complete mediation can be preserved best if policy enforcement and decision are placed as close to the resource as possible. This requires the interacting ACS to be interoperable with respect to the attribute stubs used and the security tokens that are exchanged. Even more this requires each resource domain to support the software application/system specific access control paradigm and to provide an open interface to its integrated ACS. The more monolithic an existing resource managing system is the lesser is the probability that these requirements will be met.

In these cases policy enforcement and decision must either be in the context domain or in the application domain. An example of how this could be implemented is provided in Appendix B for the example of adding access control to an existing XDS provider.

5.6.4 Separation of Access Policies and Behavior Policies

In Chapter 4, the distinction between resource access policies and resource behavior policies was made. The sample policy used in this chapter combines both aspects because it states who is allowed to access the historical database application as well as how authorized subjects might use this application (e.g., by restricting purposes and data types).

An alternative solution would have been to separate the respective policy statements by simply splitting the profile into two sentences:

*I hereby authorize **[roles]** at **[organizations]** to use the "Historical Database" Application. Access is restricted to **[Patient] [kind-of-data]** for the purpose of **[purpose]**.*

A valid instance of this template (and the corresponding policy profile) might be:

*I hereby authorize **physicians** at **Clinic A** to use the "Historical Database" Application. Access is restricted to **my lab data** for the purpose of **medical treatment**.*

For the sample scenario this split of policy purposes can be used to implement a PEP/PDP for the access part of the policy at the context domain (or at the application domain) and a PEP/PDP for the behavior part of the policy at the resource domain (see Figure 26).

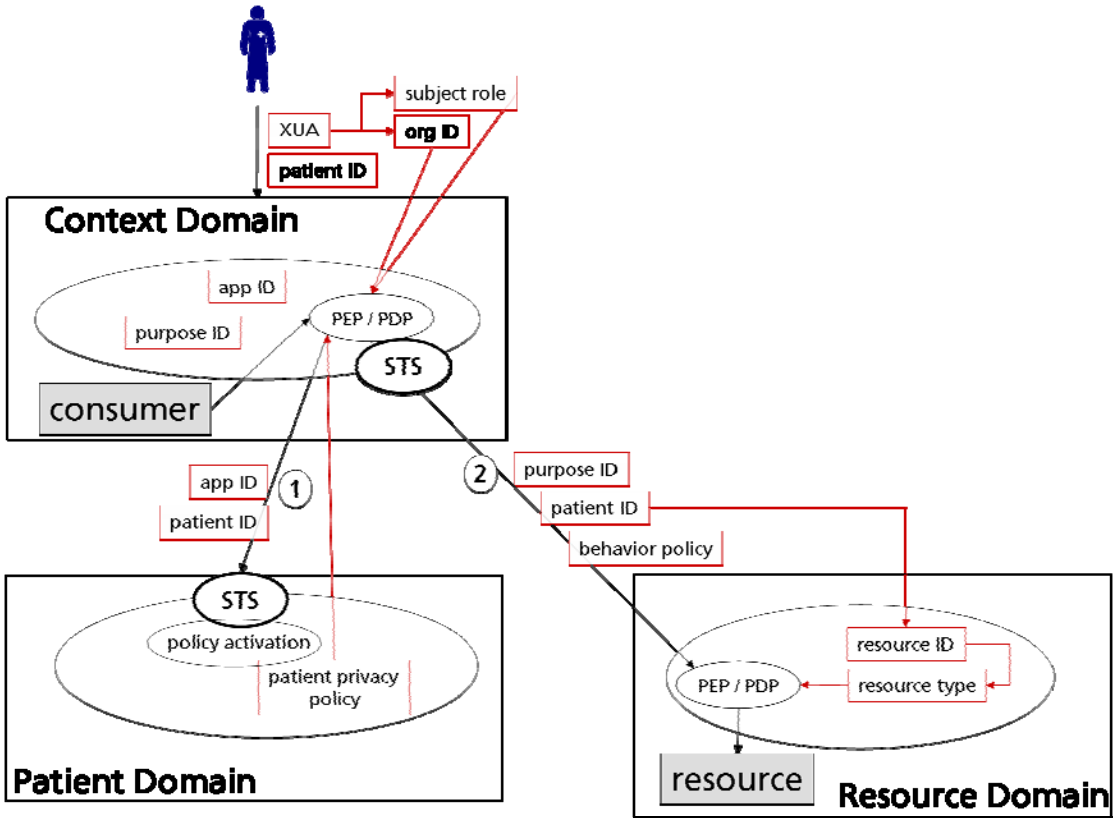


Figure 26: Separation of Access Policies and Behavior Policies

5.6.5 Policy Decision Pull, Policy Decision Push

In the previous sections it was always assumed that PEP and PDP are located within the same domain. Even if there is a good reason to place the PEP close to the resource, there might be environmental conditions and non-functional requirements causing the policy decision to move to another domain than the one where the policy enforcement is residing.

In this case, again two flow patterns can be distinguished: pushing a policy decision to the PEP and pulling a policy decision by the PEP.

In the decision push pattern is comparable to the policy push pattern with the important difference that not the policy but the result of its evaluation is pushed to the resource side PEP by the context domain. In this case the PDP might either be located at the patient domain or at the context domain.

Policy decision pull on the other hand is comparable to policy pull with the only difference that the PDP is located at the patient domain. While with policy pull the PEP-PDP communication is inner-domain and PDP-PAP communication is cross-domain, this is just the other way round for policy decision pull.

5.7 Deployment Opportunities

As described in Section 3.3.2 access control domains and physical nodes are orthogonal concepts. For instance, a client-side software application with a local logon concentrates both context and subject domain on a single node (even within a single application running on that node) while a network of registries and repositories spread among multiple hosts could be modeled as a single resource domain.

The concrete deployment of an abstract solution onto concrete nodes always depends on the non-functional requirements and existing environmental conditions that have to be considered (see next section). Nevertheless, there are some common deployment patterns that are well established and supported by existing off-the-shelf access control systems.

5.7.1 Intra-Enterprise Access Control Scenarios

The most common architecture is a variety of centralized database system for different purposes and departments (e.g., HR database for administration, PACS for managing radiologic data, Hospital Information System for workflow management and management of health records) which can be accessed by client systems. As these systems usually integrate the management of contexts, users, access rights, and resources, multiple instances of the respective logical domains exists which are mapped onto a single node for each corresponding software application/system.

The increased management efforts and limited flexibility of this architecture is mainly due to the multiplicity of logical domains rather than to the integration of domains during deployment. For instance, the introduction of a master directory for individuals, subject attributes (e.g., roles) and credentials does not replace the respective functional blocks from the managed applications. However, it extends the single application's logical subject domains into a single, enterprise-wide subject domain. Therefore, a rule-of-thumb is that management efforts depend on the number of logical domains rather than on the number of nodes these domains are deployed onto.

The same holds for the patient domain where the use of the IHE PIX and PDQ profiles does not limit application specific patient identifiers but leads to a single logical patient domain where patient specific attributes can be requested from a single, dedicated service endpoint.

Deploying context domain and resource domain onto different nodes increases the flexibility with respect to the placement of the PEP that intercepts the flow of control among these domains. If the communication protocol to access resources is known (e.g., because it is based on standard transaction) the PEP could be deployed as a dedicated node that acts as a (reverse) proxy for the resource and provides access control in a manner that is almost invisible to both the user and the resource managing system.

5.7.2 Cross-Enterprise Access Control Scenarios

With regard to cross-enterprise scenarios where fat clients are used to access either centralized or distributed resource managing systems, the following deployment does apply: The context domain is mapped onto multiple nodes (fat clients) , whereas the resource domain is mapped onto registries and repositories. It might even be a good idea to consider separate resource domains for registries and repositories. This is due to the fact that different policies might be used to control access to metadata and medical documents (see appendix A).

If IHE profiles are used for implementing a cross-enterprise health information exchange, the patient domain is deployed onto two nodes: Patient attributes (e.g., identifiers and demographics) are managed by PIX/PDQ while policies are registered and stored by XDS registries and repositories. As an XDS Affinity Domain usually only contains a single, central registry, the discovery of policies is easy. Deploying both registries (policies and document metadata) onto the same node or even using the same database for both leads to a configuration where a direct trust relationship among the policy-managing part of the patient domain and the resource domain can easily be established. In this case the retrieval of a policy using the policy pull or policy cache pattern is a local operation and therefore much more efficient in contrast to a distributed deployment of the nodes that maintain and consume the policies (i.e., policy push pattern).

The deployment of the subject domain mainly depends on the deployment of the subject attribute sources and the ability of the participating enterprises to provide these attributes in a trustworthy manner (e.g., by using XUA profile's assertion). In general a deployment should not demand that subject attributes are fetched when needed by the PEP/PDP from information points (e.g., PWP or other LDAP based directories) within the participating enterprises. Besides performance issues there might also be problems with firewall configurations, proprietary interfaces, and non-uniform key values for retrieving the required data. Therefore the preferred solutions are either deploying the subject domain completely decentralized (each participating enterprise provides all required subject attributes together with the request messages), deploying it onto a central node, or deploying the authentication part within the enterprises while managing all relevant subject attributes with a central directory. With respect to maintainability the first option will be preferred in most scenarios. If it can be used mainly depends on the result of a threat analysis because the trustworthiness of the authenticity and integrity of all subject information must match with the protection requirements of the data that is exchanged within the XDS Affinity Domain. As this issue even affects usability (e.g., if multiple logins are required), maintainability

(e.g., if multiple user accounts are needed), performance, and availability it is the most challenging when designing access control for an XDS Affinity Domain.

5.7.3 Cross-Community Access Control Scenarios

Connecting multiple XDS Affinity Domains and enabling cross-community exchange of health information leads to a new major challenge: the deployment of the patient domain.

Assuming that subject attributes are issued, retrieved and exchanged in a trustworthy manner within each XDS Affinity Domain, each XDS Affinity Domain can be represented by a single subject domain. The deployment of this consolidated subject domain is straightforward as the initiating gateway is the only node which can be used to broker trust into another domain.

The deployment of the various instances of context domains and resource domains is solely determined by the participating XDS Affinity Domains and therefore not a specific issue in the design of an access control system for cross-community health information exchange.

One of the most challenging issues when connecting XDS Affinity Domains is the alignment of the single XDS Affinity Domains' patient domains and the linkage of a particular XDS Affinity Domain's patient domain with context and resource domains of other XDS Affinity Domains:

- The patient is known in multiple XDS Affinity Domains which might use different patient identifiers and attribute sets. e.g., a resource security policy in XDS Affinity Domain A relies on the kind of health insurance the patient has. This information is managed by the patient domain of XDS Affinity Domain B. The context domain is located in XDS Affinity Domain C.
  In most cases the most efficient solution is to use XCPD Integration Profile to resolve the patient identifier in the responding community and to retrieve additional attributes on the patient. If attributes are not synchronized and therefore not known, an attribute service that is located in the patient domain should be used to collect patient attributes from different patient domains. This achieved by querying the patient identity sources of these domains.
- The patient's privacy policy might be hosted within any, or all, of the participating XDS Affinity Domains. In most cases either the context domain or the resource domains, or both have knowledge on how to discover the patient's privacy policy. Therefore, it makes no sense to design for a policy push if the required attributes for policy discovery are solely available at the remote resource domain. To solve this issue, the clear rules for the discovery of policies must be defined.

Cross-community access control gets hard to implement if the various XDS Affinity Domains use different taxonomies (e.g. role sets) for the definition of patient privacy policies and corresponding attribute stubs. While policies are consistent with the subject attributes issued in the same XDS Affinity Domain, it will require additional semantic matching to evaluate a policy against attributes that were defined within another XDS Affinity Domain. The only solutions for this are to either agree on a common set of attribute stubs and value sets or to use semantic technology for mapping and matching controlled vocabularies from multiple communities.

6 Recommendations to IHE – Actors and Transactions

This section introduces all necessary Access Control System actors and – where relevant – associated transactions. It should be noted, that the described actors are not restricted to certain domains (see Sections 3.3.2 and 5.4.). They are rather general and might used in different instances in more than one domain.

6.1 Security Token Services

Access control information (e.g., subject attributes, policies) which is exchanged between ACSs, must be safeguarded by encapsulating them into security tokens. Encapsulation enables any involved ACS to attest the trustworthiness of each piece of information. The desired outcome is a chain of individually approved and comprehensible pieces of security-related information, which – combined to each other – form the technical representation of the mutual trust relationships of all participating systems.

A security token exchange scenario consists of those actors:

- An X-service provider (business level actor and respective ACS) who requires authentic claims on a certain system entity in order to verify the legitimacy of an access to a managed resource.
- An X-service user who needs to provide authentic claims about a system entity to an X-service provider, in order to access a protected resource.
- A security token (ST) provider, who asserts authenticity of claims. The contents of the claim – and even the mechanisms for safeguarding it – may be further refined by a specific profile in order to support additional requirements (e.g., inclusion of a proof-of-possession attribute).

Figure 27 shows these actors and the transactions among them.

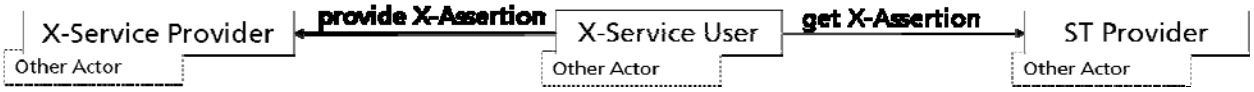


Figure 27: Security Token Exchange Actors

X-Service User and ST provider are abstract actors from which other actors can be derived that require a trusted encoding of their issued claims (comparable to the publisher and subscriber actors).

Recommendation: IHE should define a framework for the definition of interoperable "get X-Assertion" and "provide X-assertion" transactions. This framework should consider two different levels of trust: direct trust (X-Service User consumes X-Assertion) and brokered trust (X-Service User as intermediary between X-Service Provider and Security Token Provider). For a discussion on how the actors and transactions shown in figure 27 can be mapped onto existing standards see appendix C.

Specific problems regarding security token sharing or exchange may arise in cross-community scenarios. The whole concept of trust brokerage relies on the ability of the X-Service Provider to verify that an X-Assertion is authentic and that is was issued by a trustworthy ST Provider.

In cross-community scenarios this ability might not be available (e.g., because an XDS Affinity Domain does not provide a way for external users to verify it's internally used digital service certificates) or not even be desirable (e.g., because an XDS Affinity Domain wants to hide its internal setup). In these scenarios the gateways that broker the messages among the XDS Affinity Domains must also broker the trust relationship (Figure 28).

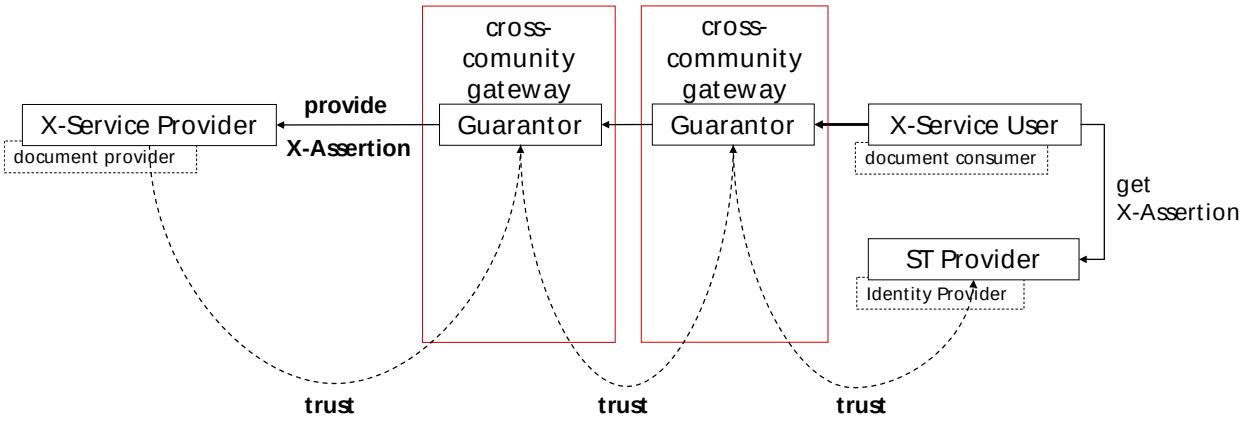


Figure 28: Cross-Domain Gateways for Brokered Trust Establishment

Recommendation: IHE should provide guidance on how XUA can be used in cross-community scenarios based on XDS, XCA and XCPD without having to bridge or connect both communities' PKIs.

6.2 Attribute Provider and Attribute Consumer Actors

An attribute provider actor is derived from the abstract ST provider actor. Attribute consumer actors query an attribute provider actor for attribute values on a defined entity by providing the unique identifier of this entity (white pages semantics).

Attribute provider actors implement policy information points that can be queried by a PDP (which takes the role of an attribute consumer actor) in order to retrieve attribute values that are needed for the evaluation of a policy.

Recommendation: IHE should define an attribute provider (semantic of a policy information point) for querying attributes about objects (i.e., subjects, patients, and resources). The respective actors and transactions are needed for infrastructures where e.g., subject authentication is performed within a domain that does not maintain role and organization membership information about the authenticated subject (e.g., if a health professional card is used) authentication is performed within a central subject domain (e.g., a nationwide PKI) while most of the subject's attributes are managed with the enterprises subject domain (e.g., enterprise HR services).
For a discussion on how the actors and transactions of an attribute service can be mapped onto existing standards see Appendix C.

6.3 Identity Provider

An identity provider is derived from the abstract ST provider actor. Its functionality is to provide identity consumers with authenticated subject information on an identifiable subject. Security tokens issued by an identity provider may include mechanisms that enable an identity provider consumer to verify that the authenticated subject information is about the subject that hands over the token. The IHE XUA integration profile defines the specific instance of the "provide X-assertion" transaction to exchange authenticated subject information among a service user and a service provider (see appendix C for further details).

6.4 Policy Activation and Retrieval

In Section 5.6 we sketched three different patterns for exchanging policies among ACS in the affected domains (context domain, patient domain, and resource domain). While the policy pull pattern is based on direct trust among the patient domain ACS and the policy evaluating PDP, the other patterns rely on brokered trust.

This requires that the policy evaluating PDP trusts the client (context domain) to provide an authentic, unmodified policy that may be fully verified and validated by the access control scenario. Given the assumption that the context domain is the hardest to protect against intruders, attacks must be considered where a formerly intercepted (or manipulated) policy is re-issued for another scenario. This kind of attack can only be prevented by either encoding the access control scenario within the policy or by accepting policies only in conjunction with a safeguarded claim that a certain policy was activated for a certain scenario.

Whereas the first countermeasure may only be applied at policy design time, a dedicated activation claim is verifiable even at run time. Therefore these countermeasures should not be considered as alternatives but as complementary security measures.

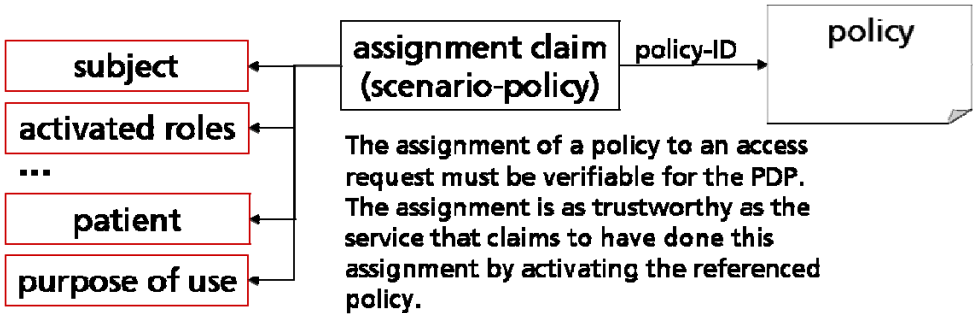


Figure 29: Policy Assignment

Based on this consideration, two transactions are required to implement all three policy activation patterns:

- Retrieval of an assignment claim (policy activation for a certain scenario)
- Retrieval of a policy for a given assignment claim (policy-ID)

The policy activation transaction that provides a policy claim can furthermore be used to implement implicit policies where the PDP "knows" from the policy ID what rules have to be enforced.[8]

For direct trust scenarios (e.g., policy pull within a single XDS Affinity Domain) both of the sketched transactions map onto "ordinary" registry and repository transactions, where the assignment claim corresponds to the identifier of the respective policy document.

For brokered trust scenarios (e.g., cross-domain, policy push, policy cache), the authenticity of the issued assignment claims must be verifiable for the PDP. Therefore the issuing actor should be derived from the previously defined "ST provider" actor. Depending on the activation pattern the policy enforcing PDP and the ACS of the context side business service consumer take different roles:

Pattern	Policy Activator	Policy enforcing PDP	Context domain ACS
Policy pull	ST Provider	X-Service User as X-Assertion Consumer	(not involved)
Policy push	ST Provider	X-Service Provider	X-Service User as X-Assertion Broker (claim)
Policy cache	ST Provider	X-Service Provider (claim) and X-Service User (policy) as X-Assertion Consumer	X-Service User as X-Assertion Broker (claim)

Nevertheless it should be noted that policy pull scenarios within a single domain usually rely on direct trust and therefore do not require specific means for safeguarding claims and policies.

Figure 30 shows how the respective actors interact in order to implement all three patterns for both direct trust and brokered trust scenarios.

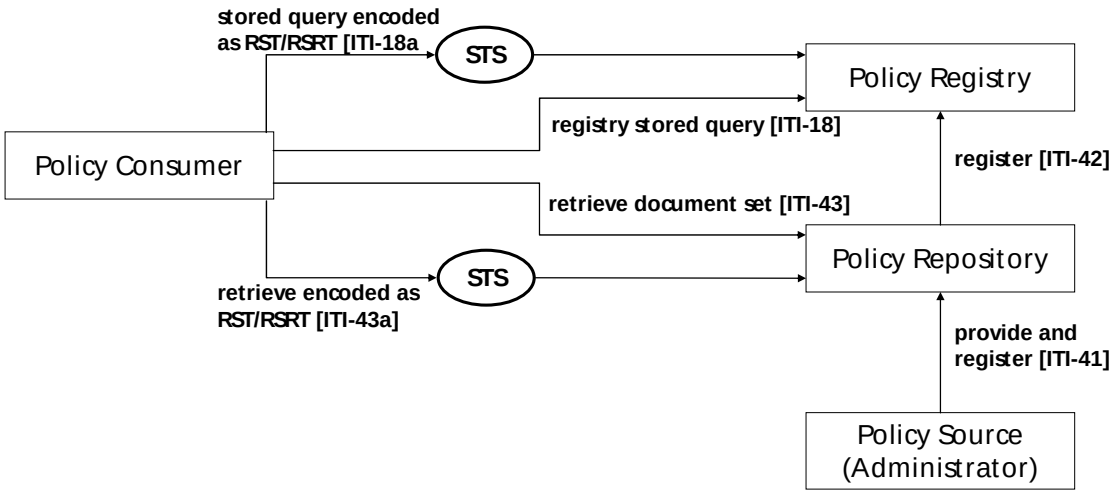


Figure 30: Policy Actors

[8] This mechanism can be used if policies are not machine-readable. In this case certain policy semantics can be assigned unique identifiers and the policy activating actor will just select which semantic applies for the current scenario. The PDP must then either hard-code these semantics or use proprietary mechanisms to obtain the matching rule set.

Recommendation: IHE should provide guidance on how to use XDS stored queries and existing document retrieval transactions to implement dedicated transactions for policy activation (claim retrieval) and policy retrieval. For direct trust scenarios such guidance is already given in the BPPC profile, but this doesn't cover brokered trust scenarios. For a discussion on how the actors and transactions shown in Figure 30 can be mapped onto existing standards see Appendix C.

6.5 Role Activation

The context-specific activation of the subject's current role(s) can occur in five different ways (see Section 5.5.1 for examples of the first two opportunities):

1. An identity provider actor within the subject domain provides claims on the activated roles of the subject as part of the authenticated subject information. This requires that all information needed for role activation (e.g., context identifier, purpose of use) is made available at the subject domain as part of the "get X-assertion" transaction. This restricts the lifetime of the XUA assertion to the time span the subject is within the current context.
2. The set of all applicable subject roles is filtered within the context domain and only claims on the currently activated roles are forwarded to the business actors ACS as part of a "provide X-assertion" transaction. In this scenario the context domain takes both the roles of a "ST provider" actor and a "X-Service user" actor, which requires the context domain to provide the respective security context and level of trust.
3. The PDP requests claims about the activated subject roles from a PIP that is located within the subject domain. This requires that all information needed for role activation (e.g., context identifier, purpose of use) is made available at the subject domain by the PDP (X-Service user) as part of the "get X-assertion" transaction. As an alternative solution a role activating PIP can be located within the context domain and provided with the set of all applicable roles by the PDP (who retrieved this information as part of a provide X-assertion transaction). Both approaches require at least one additional message and a close harmonization of the interfaces and the attributes of the services representing the context, subject, and resource domains.
4. In scenarios with a direct trust relationship among the context domain and the resource domain, role activation can be implicitly handled by policies. By formulating policy rules on static administrative or organizational roles (encoded within an XUA assertion) and on context attributes (provided with the message) there is no need to match roles and context in advance at the context domain.
5. Role activation is implicitly handled by a local security policy that is enforced by a context domain PEP. Instead of mapping functional roles onto permissions this policy works on couples of static administrative/organizational roles and context attributes (e.g., current task). It is responsible for blocking all messages and medical data which are not accessible for the user's static roles within the given usage context. Using this option all policies that are enforced at the resource side solely decide on access rules based on organizations, individuals, and/or static role assignments.

Recommendation: IHE should not define specific means for explicit subject role activation. An implicit role activation as proposed by options 4 and 5 should be preferred because this can easily be implemented by using existing standards and profiles.

6.6 Policy Enforcement and Policy Decision

The interoperability issues of the policy enforcement and policy decision actors can be kept to a minimum if:

- There is no PEP-PDP communication across XDS Affinity Domains; for most deployments this can be assumed as easily achievable. The only exceptions are deployments where – e. g. because of the localization of certain attribute sources or because of the confidentiality of certain attribute values – the evaluation of a policy cannot be done within the resource domain.
- Either all XDS Affinity Domains agree on a common standard for policy encoding, provide redundant PDPs for all standards used, or always decide on a policy within the XDS Affinity Domain where that policy is managed.
- Policies are managed by XDS registry/repository and existing XDS transactions are used for the retrieval of policy IDs and policies (see section 6.4).

As shown in this paper, the integration of a PEP into the flow of control is rather an architecture design issue than an interoperability problem. Therefore any attempt to cover this by standardized transactions will fall short because of the diversity of messages one could think of as candidates for being intercepted.

Recommendation: No additional actors/transactions are needed for single XDS Affinity Domain scenarios. For multiple XDS Affinity Domain scenarios the interoperability issues can be reduced to PEP-PDP communication and the syntax of encoded policies. Due to a close integration of these building blocks by existing products there seems to be no urgent demand for normalization. Therefore IHE should take a pragmatic approach and provide a white paper or cookbook on how to integrate a PEP/PDP into the XDS flow of transactions assuming a close integration of PEP and PDP.

7 Appendix A: Requirement Specific Component Deployment

Chapter 5 was introduced with the statement that depending on non-functional requirements and environmental conditions the same use case will be mapped onto different ACS deployments and flows of control among ACSs. In this section some examples are given that all build upon the same sample scenario but assume specific environmental conditions and requirements.

7.1 Thin Client

Environment and Requirements:

Only web browsers are to be used as clients.

Possible Solution:

The context domain and its ACS are deployed as a web portal that is accessible through a web portal. Health care providers (HCPs) use a single sign-on with a second web portal. Whenever the web portal is called without a valid assertion (as defined by XUA) the portal redirects the user to the log-in portal. The log-in portal verifies the user's credentials and upon success redirects the user back to the web portal. This interplay of the two portals is invisible to the user and can be automated by using standard protocols (e.g., SAML HTTP POST Binding).

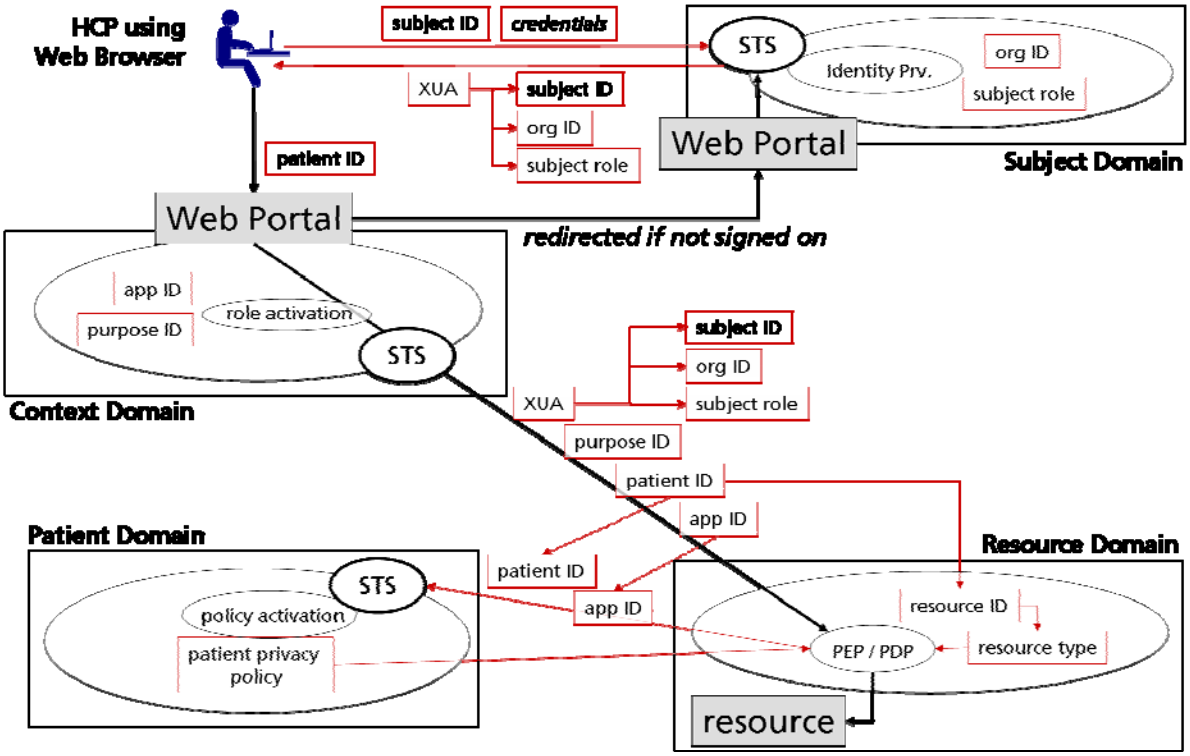


Figure 31: Optimization with Thin Client

7.2 XDS Affinity Domain

Environment and Requirements:

The hospital network is set up as a XDS Affinity Domain with a central registry and distributed repositories within the participating hospitals. Queries for patient historical data are based on the IHE ITI-18 Registry Stored Query transaction. The registry's response only contains references to documents that match the patient's privacy consent. Data itself is retrieved using the IHE ITI-XDS.b Retrieve Document Set transaction. Again the restrictions of the patient's consent have to be considered.

Possible Solution:

The registry and all repositories are modeled as resource domains. Each resource domain contains its own ACS which enforces the patient's privacy consent. In contrast to the previous example a policy push pattern is recommended because otherwise the policy would have to be discovered and fetched for each repository access.

It has to be noted that the policy has to be evaluated and enforced for each query and retrieve transaction. Assuming that the registry is only queried once this results in n+1 evaluations of the same set of rules against the same set of attributes for the retrieval of n documents.

7.3 Document Prefetch

Environment and Requirements:

When a patient registers for a physician visit the respective organization may want to fetch that patient's historical data in advance. This results in a second copy of the data stored at the requesting organization. This second copy must also be protected from illegitimate disclosure with respect to the access rights granted by the patient.

Possible solution:

This scenario can be implemented by introducing an additional (local) resource domain, in which the prefetching software application/system is located and operated. Policies are pulled by the prefetching application and then pushed forward to the remote resource domains. In order to simplify the evaluation of the consent, the template is reordered and split into two statements:

*I hereby authorize **[organizations]** to use **[Patient] [kind-of-data]** through the "Historical Database" Application. Within [organizations] access is restricted to **[roles]** for the purpose of **[purpose]**.*

Using this reordering, the first statement legitimates the prefetching operation while the second one can be used to verify the legitimacy of a subject's access to the local copy at the user's site.

Figure 32 shows the flow of data and control for the first step of this scenario, the prefetching of all of the patient's data that matches the first consent statement. Due to the complexity of the figure each step will be discussed separately:

Prefetch:

The functional blocks that are responsible for prefetching patient data are modeled to act as a dedicated resource domain. The business functionality within this domain consists of a prefetcher software application/system and a data store. It is assumed that this data store is physically mapped onto the local data store of the organization that requires the data. Access to both, the

prefetcher and the local data store, is safeguarded by a local ACS which incorporates a PEP/PDP system that intercepts any call to the business functions.

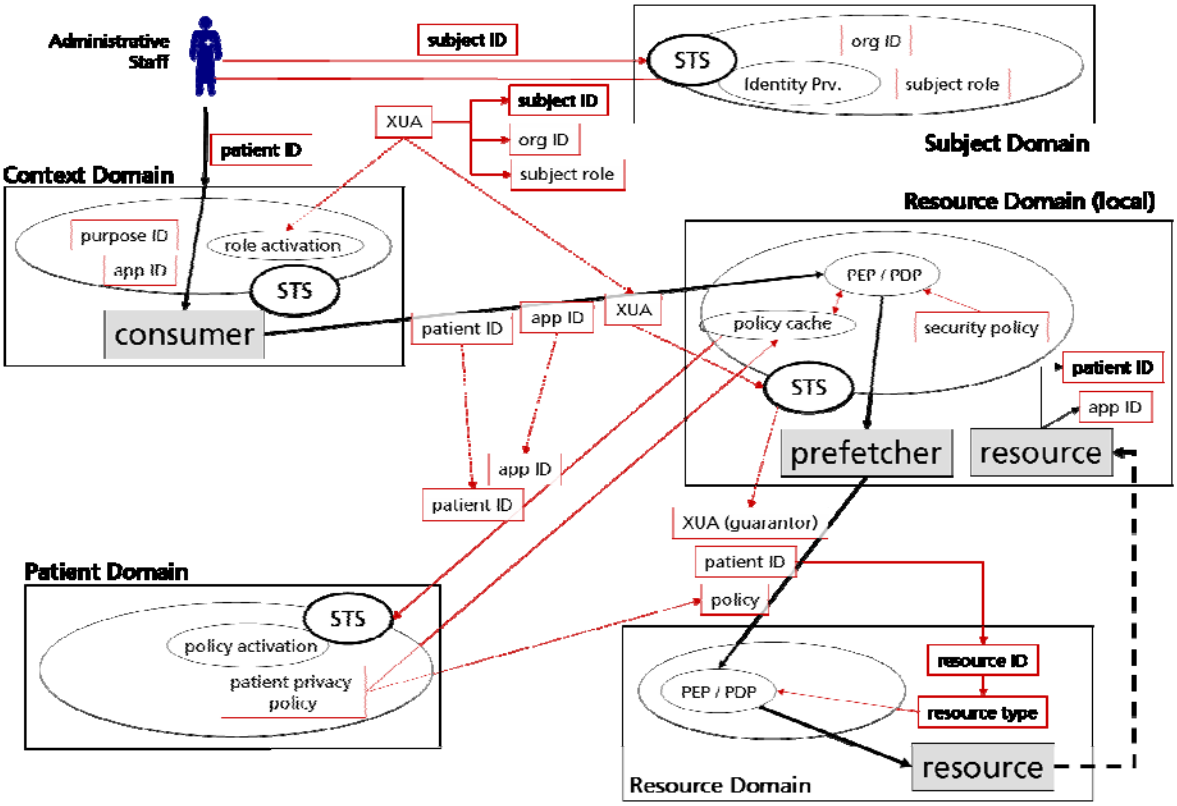
It is assumed that prefetching is initiated by an administrative staff member (e.g., as part of the preparation of a visit after a date has been fixed). Using the local IT system (e.g., a hospital information system), an administrative staff member triggers the prefetcher by transmitting the identifier of the patient and other descriptive data such as the date of the visit. The prefetcher requests the patient's historical data from a remote source and stores it in the local resource domain's data store. A physician is then able to retrieve this data using a local operation.

Provider Access:

The administrative staff member logs in as usual (e.g., using a single sign-on as shown in figure 32) and retrieves an XUA assertion as a proof of successful authentication. The XUA assertion carries information on the subject's role and organizational membership. While initiating the prefetch operation the XUA assertion is forwarded to the local resource domain. Local application logic at this domain verifies the permission of the subject to initiate a prefetch operation. This is done by using a PEP/PDP and a local security policy that states which roles are allowed to prefetch data. After successful authorization, a local PEP encapsulates the XUA assertion with a security statement that attests the successful authentication and authorization.

In order to fetch the policy only once, the local resource domain then fetches the patient's privacy policy from the patient domain and temporarily holds it within a local policy cache.

This guaranteed assertion (namely the confirmed XUA assertion mentioned above) and the policy are then forwarded to the remote resource domain. The PEP/PDP within this domain is able to verify the signature of the prefetcher and therefore does not have to verify the authentication again. Its only function now is to verify the first statement of the policy against the organizational membership of the caller and - on success – returning the required medical data objects to the prefetcher.



1895

Figure 32: Document Prefetch with Single Sign-On

After the data has been prefetched it is accessible for the eligible physicians. For every access attempt, the second statement of the consent is evaluated. The respective policy is discovered using the patient identifier. It does not have to be fetched again because its use was already
1900 anticipated and the policy is subsequently held at the local resource domain's policy cache.

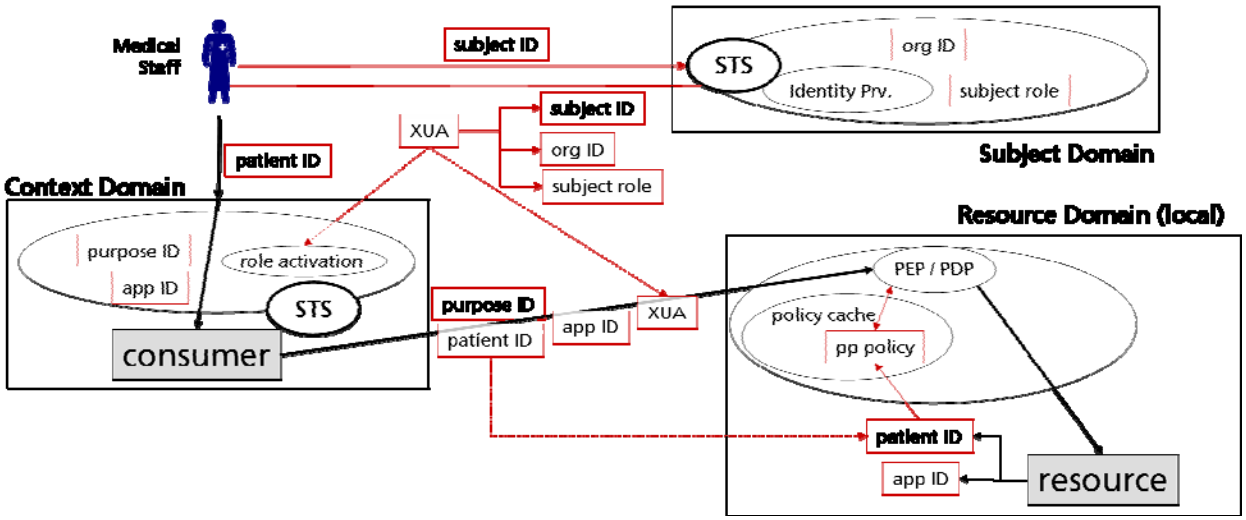


Figure 33: Using Previously Prefetched Documents

7.4 Multiple XDS Affinity Domains

Environment and Requirements:

The existing hospital network is joined with two other adjacent regional networks, with each of the three networks being set up as an independent XDS Affinity Domain. Patients can give consent that each physician connected to this extended circle of trust may access historical data even cross-domain. This consent is given with each XDS Affinity Domain where historical data of that patient is gathered.

Possible solution:

In this scenario each registry and repository within any of the XDS Affinity Domains represents a resource domain on its own. Cross-domain messaging and routing is implemented using the IHE XCA profile and therefore the document consumer does not have to be aware of the distributed nature of the historical data application. Policies have to be enforced within each resource domain. Due to the dedicated consents these domains are responsible for the legitimacy of data processing. Policies can only be pulled because they are distributed among the XDS Affinity Domains and therefore only a node within a domain can "know" where the respective policy is made accessible.

In order to operate that scenario, all affected XDS Affinity Domains must agree upon the semantics of the attributes that are used to express roles and other subject properties. Otherwise a resource-side PEP/PDP is not able to use these attributes for the retrieval and/or evaluation of the patient's privacy policy, which may obviously result in an access denial (following the principle of fail-safe defaults).

8 Appendix B: Example - Electronic Case Record

The electronic case record (eCR) is an application that allows the seamless exchange of medical data among organizations. The specific characteristics of a case record incorporate a means to provide a virtual integration of distributed data that belongs to the treatment of a single indication of the patient. Therefore, a case record implicitly features a dedicated purpose of use and a limited set of users (organizations and individuals involved in the treatment of the dedicated disease).

The eCR application features may be summarized as follows:

- The application defines three roles: (1) a record initializer, who creates new case records, (2) an editor, who processes medical data within a case record, and (3) a case record manager, who is in charge for the compliant operation of the case record and who acts as the patient's primary contact for all administrative issues. Case record providers may introduce additional roles for special use cases (e.g., the role of a quality supervisor if a case record is used to support specific managed disease contracts).
- Only data that a person with the role of an editor has declared relevant for the respective disease is linked into a case record
- The patient has to state his/her consent towards the organizations and individuals that might access his case records (one consent per case record). The patient may withdraw this consent freely at any time for either a single accessory or for the whole case record.
- For each case record role, a policy is provided, which controls the behavior of this role, e.g., each editor can access all data within the record while other (future) roles might only access objects of certain type.

Each identity provider that is used for creating authenticated subject information within participating organizations, adds specific attributes to the respective security token, which allows a clear assignment of the subject to one of the case record roles. When an authenticated user requests access to a case record these attributes are evaluated and the user is assigned a machine readable policy that reflects the access rights of the respective case record role. This policy is communicated alongside with any request to a case record service, which is responsible to enforce and decide the policy before accessing any protected resource. By using predefined policy profiles for expressing the roles' permissions, the application semantics is translated into role-specific resource behavior policies. These policies merely define what the policy subject is allowed to do with the case record, but they do not explicitly restrict access to the case record as a whole.

The patient's privacy consent is managed similar to the concept of an access control list for each individual case record, in which all individuals and organizations are listed that are allowed to access a certain case record. Case record roles and the respective policies for a certain case record are only relayed to subjects that are listed in this case record's ACL. If organizations are listed in the ACL, a specific subject attribute that states the organizational memberships of the individual is evaluated. Therefore the patient's consent is enforced solely as a resource access policy. The patient has to agree on the application semantics and cannot influence the resource behavior.

Compliance and organization of labor affect the use of case records at two points:

- The participating organizations agree on the assignment of case record roles to their internal roles. For instance, in a typical configuration patient admission staff is assigned the case record role of the record initializer while physicians act as editors. If a case record is used to support a managed care contract usually the care manager takes the role as the case record manager, too.
- The case record semantics relays the decision on who is involved in the patient's treatment to the organization of labor within the participating organizations. Therefore, these organizations have to enforce policies that ensure that only individuals that need to access the patient's case record are able to log in to the case record application and to open the respective record instances of a patient.

A further mechanism of the case record is the so-called offline token (e.g., a plastic card or a referral letter with a barcode printed on). The patient may utilize that token in order to explicitly grant access rights for one of his case records to a physician. Each offline token is bound to an access policy and a behavior policy:

- The (optional) access policy contains rules for deciding whether the physician is allowed to use the offline token. A common use case is to restrict the use of an offline token to certain medical specialties, e.g., for enabling the patient to set up a care team of dedicated roles which are defined by a managed care contract
- The behavior policy defines which permissions are assigned to the person that uses the offline token. This not only includes the definition of allowed operations on resources but may also feature restrictions on the time of validity of the token (e.g., one-time access).

Due to the clear purpose of use and the strict separation of policy concerns, there are no policy conflicts and all policies can be evaluated in a sequential manner at the dedicated points within the flow control. Another opportunity of managing these policies is the design of special items (in fact policies) that are combined to a policy set that finally is evaluated as one whole policy. However, this would require the consistent use of one policy language.

9 Appendix C: Implementation Issues (incl. Standards and Profiles)

The service-oriented approach for the presented access control system reveals much complexity when implementing actors and transactions. However, there exist well-established standards and best practices in order to track interoperability in message security. The introduced standards are not mandatory for use, but they illustrate the core concepts in a reasonable way.

9.1 Overview of Security Standards Composition

As mentioned before, there exist several security related standards in the field of Web services and XML. The following diagram show the interaction of such standards and what they are designed for.

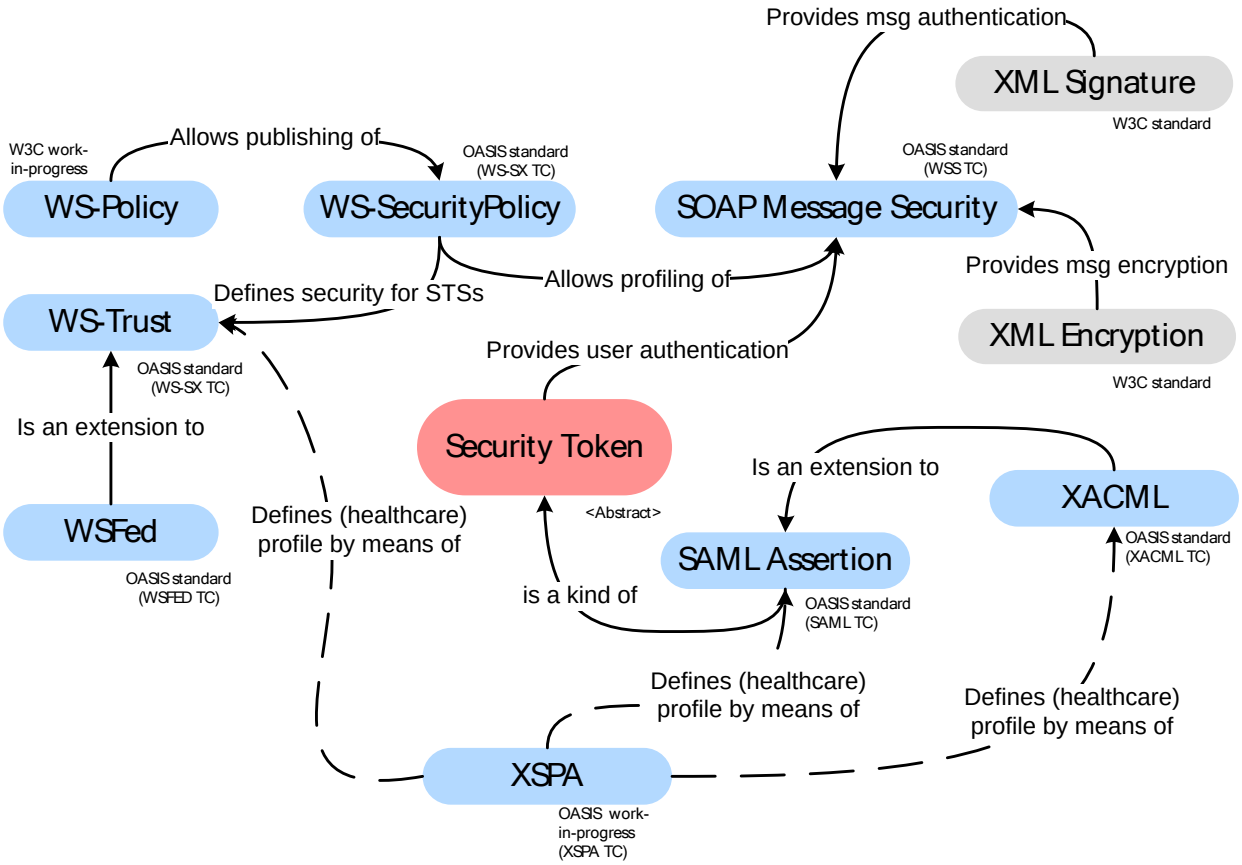


Figure 34: Related Security Standards

9.2 Layering Opportunities (Message Header, Payload)

By using Web services, various locations inside the SOAP message with regard to carrying security aspects are possible. SOAP provides flexible mechanisms of extending a message. The SOAP header serves the transmission of data that does not belong to the formal message signature. For instance, security token, signatures and transaction identifiers are popular candidates of such data. Nevertheless, it is not banned to use the SOAP body. The following

diagram shows a standard way of placing a security token (i.e., SAML assertion) into the SOAP security header.

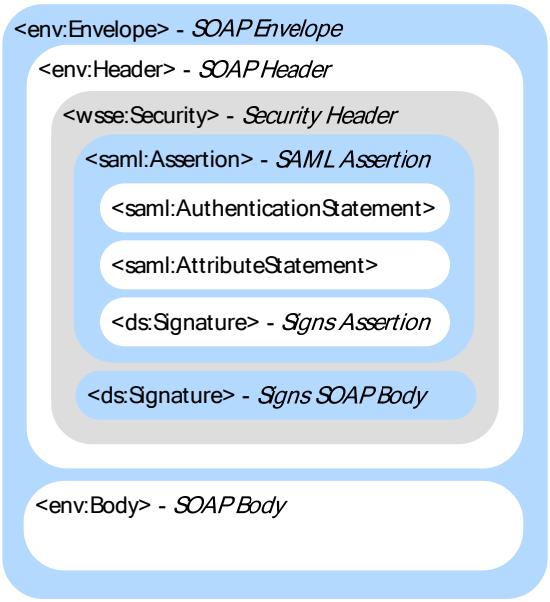


Figure 35: Placement of SAML Assertion into SOAP Security Header

In order to ensure that the SOAP header block is understood and processed mandatory, there exist the *mustUnderstand* attribute which must be set to *true*. By contrast with the SOAP body, Web service implementations are indicated a fundamentally place of processing the SOAP message. The standard WS Security defines an additional security header inside the SOAP header. The security header inherits the *mustUnderstand* attribute, what enables a message sender to accomplish the underlying security demands of SOAP nodes. For interoperable reasons, it is recommended to use the security header for additional security data. Indeed, the SOAP specification does not deny any extensions, but user-specific extensions may not be processible by all participating services (i.e., federated services).

9.3 Security Token Encoding and Exchange

The interaction of the access control systems or reference monitors needs the delegation of a security context. For instance, security policies, (activated) roles and authentication credentials are part of a context. A security token refers to the embodiment of such security objects. They can be encoded in SAML end exchanged by means of the protocol specified in WS Trust.

9.3.1 SAML and WS Trust

WS Trust is specified as an enhancement of WS Security in order to formulate and process security token in standardized way. It establishes a trustworthy communication. The underlying trust model defines three roles: requestor, Web service, and security token service. The Web service externalizes e.g., the authentication processes, and thus delegates the security functionalities to a trusted security token service. The security token service is able to identify principles known to the system and associate them as subjects. A requestor that provides

authentication credentials (e.g., a password or another security token) is issued a security token that asserts the use of a trusted service.[9] The requestor can provide this token in order to execute the desired operation of the Web service.

Although SAML supplies an own protocol for authentication and authorization issues, only so-called assertions matter. An assertion is an instance of a SAMLToken (or IssuedToken with assertion type) that WS Trust defines and can pick identity information and various attributes. An important feature of assertions is the opportunity to state a subject confirmation method. This enables a *proof-of-possession* mechanism when using the holder-of-key confirmation method. The relying party can verify both the (authenticated) identity and the eligible purchase of the assertion. In general, SAML assertions are transferred in the SOAP security header (except the issued response message).

9.3.2 Subject Authentication and Subject Attribute Exchange based on XUA (Protection Token)

The XUA actors (X-Service User, X-Service Provider, and X-Identity Provider[10]) can represent WS Trust actors. Subject authentication is carried out by an identity provider. In addition to that subject attributes (may include roles) are provided by this provider, too. The source of the attributes is out of scope (e.g., a (federated) directory service). The result of the authentication and the attribute assignment is an XUA assertion. XUA does not make assumptions for accepting additional attributes in the assertion (this is referred to the SAML specification itself).

The XUA assertion can be used as a protection token. With regard to WS Security Policy, a protection token protects the communication key between requestor and provider that is used for signing and/or encrypting requests and responses. The service provider therefore defines a bootstrap policy that must be fulfilled by requestors (X-Service Users). The policy must specify what kind of protection token is acceptable, in this case a XUA assertion. It is necessary to issue a holder-of-key assertion by an identity provider. The key in the assertion serves as a proof key and a protection key at once.

9.3.3 Encoding and Exchange of Policy References and Policies as Security Tokens (Supporting Token)

Signed assertions may cover the encoding and exchange of policy references and policies, too. Policy references and policies can be treated as attributes and therefore be added in attribute statements. The assertion represents a supporting token in contrast to the protection token. There is no need to sign the message again, since the policy provider still signed the assertion. It must be made sure that the declared subject in the XUA assertion (protection token) corresponds to the supporting token's subject carrying the policy references or policies.

[9] WS Trust differentiates four bindings to process security tokens: issuance binding, renewal binding, cancel binding, and validate binding. [Reference: Anderson et al.: Web Services Trust Language (WSTrust). Version: February 2005.]

[10] The XUA profile does not prescribe the implementation of an X-Identity Provider. An identity provider can be implemented as a SAML identity provider or as a WS Trust security token service.

9.3.4 Security Token Chaining

If more than one security token service comes into play, issued security tokens are likely to build up on each other. This requires a (semantically) linkage between these tokens in order to verify a correct chaining. Establishing a security token chain depends on the token's type. If XML-based tokens are used, this can be easily accomplished.

A SAML-based token (i.e., assertion) allows for referencing or embedding other assertions by using the elements "Advice" or "Evidence". This mechanism can be used to create a connection in the chain. In this case the assertion identifier is the key that embodies the reference. If an assertion is issued many times, the linkage could be established by creating a digest all of the data constituting an assertion (subject, attributes, etc.). This hash is appended as an attribute. Assertions that have a reference to another assertion just add this hash. Due to the fact, that all assertions are likely to be digitally signed, the chain can be verified by equal hashes.

For instance, an XUA assertion might be bound to another (policy) assertion that holds or references an access policy (see section 6.6.4). Such a policy assignment to a subject and/or activated roles can be validated by means of the security token chain.

9.4 Policy Encoding and Retrieval

When modeling security access control policies for an organization, the underlying access control paradigm, and thus the kinds of policies (e.g., discretionary policy or mandatory policy), have to fit to supporting policy languages. In order to accomplish interoperability in distributed access control systems, a standardized request/response language and encoding of the policies itself must be present. However, this does not eliminate the option to use an own/proprietary policy format, but this must be transferable into (or injective for) a commonly accepted format.

This section introduces OASIS' XACML and ContentGuard's XrML as candidates for policy encoding. XrML provides a compact policy language to express rights to all kinds of resources including trusted content as well as services. Whereas the protocol between PEP and PDP in the SAML specification is very simple, the XACML specification (as an extension to SAML) provides much more functionality to express both simple and complex requirements to policies (e.g., obligations, environments, etc.).

9.4.1 XACML

The eXtensible Access Control Markup Language is both a policy management specification and a policy framework. It provides a policy language data model and a request/response protocol.

Each policy consists of one or more rules. To avoid competing rules, a rule combination algorithm is denoted (such as "deny-overrides"). Each rule permits or denies access to targets and can be extended with optional conditions. Policies and rules must match to a target. In this case a target can be a subject, resource, action, or an environment. This is defined by predefined or user-defined attribute identifiers, which must correspond to the ones from authorization decision requests. One important feature in XACML is the support of nested policies. This enables you to specify independent policies which can be injected into a subject bound policy set.

9.4.2 XrML

The eXtensible Rights Markup Language is another opportunity to express access rights in XML. Although the specification is initially intended for digital rights management, it might be used for access control management in healthcare, too.

A policy refers to a license that can be constructed using mandatory and optional terms. The core data model consists of four entities: principal, right, resource, and condition. A license encapsulates these entities into grants in conjunction with one or more license issuer. A comfortable feature of XrML is the possibility to digitally sign a license by the issuer. Moreover, a joint signing is supported so that e.g., a license is applicable to more than one XDS Affinity Domains.

9.4.3 Policy Retrieval

All three policy activation patterns (policy push, policy pull, and policy cache) require two transactions in order to implement the policy retrieval (see section 6.4):

1. Retrieval of policy assignment claim (policy activation for certain scenario)
2. Retrieval of policy for given assignment ID (policy ID)

The implementation of these transactions depends on the relying trust relationship between the policy consumer and policy registry/repository and the activation pattern. In case of brokered trust and policy push, the assignment and the policy must be trustworthy. Thus, both the policy registry and policy repository act as a security token service and issue the data encapsulated as a security token. Such security token services might add the assignment and the policy itself as an attribute in a SAML assertion. Since security token services are necessary for policy retrieval, the WS Trust protocol should be used to integrate these actors. WS Trust specifies the *RequestSecurityToken* message which holds for the requests of policy retrieval.

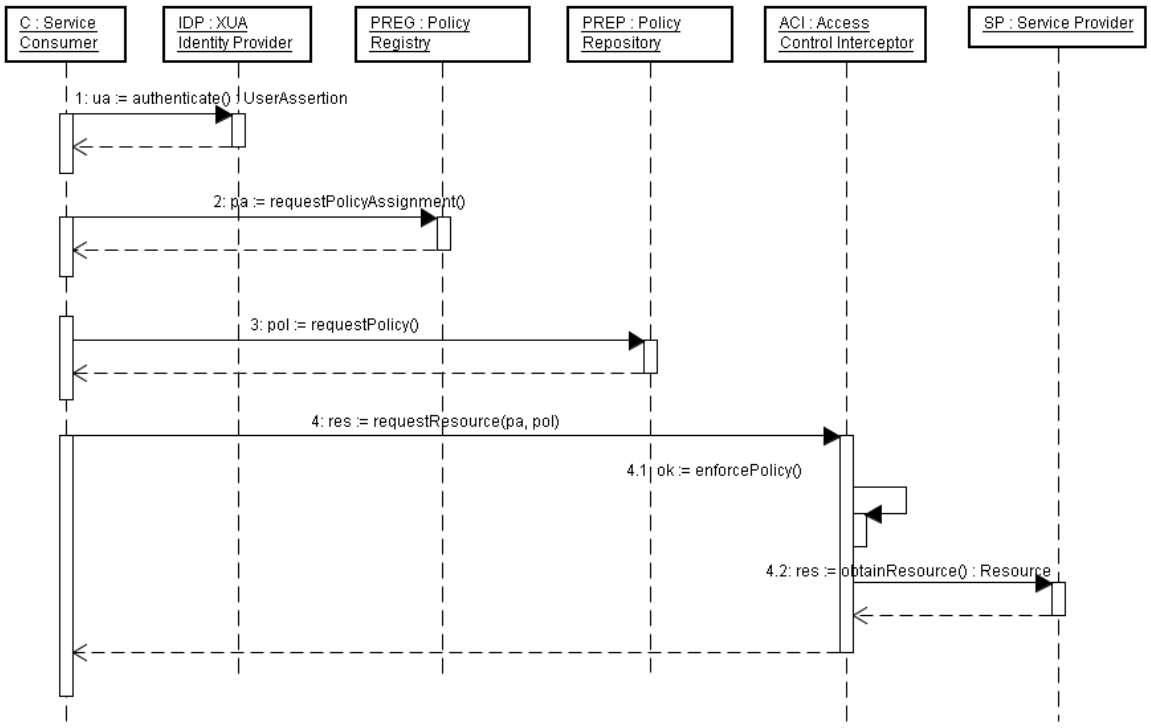


Figure 36: Retrieval and Enforcement of a Policy

In case of direct trust, the BPPC integration profile with the specified actors and transactions might be used to implement policy retrieval. The Document Consumer actor can query an XDS Affinity Domain for policy assignments and policies.

9.5 Attribute Retrieval

As mentioned in section 5.4, different kinds of attributes must be available in the core domains. Each domain must be able to provide specific attributes. There exist various attribute repositories that can be used to fulfill this requirement. For instance, the *IHE Personal White Pages (PWP)* profile and *IHE Patient Demographics Query (PDQ)* are such candidates. If there is a need to identify a patient e.g., (future) electronic reimbursement purpose of a health insurance, the demographics data (e.g., name, date of birth, address, and phone number) can be used to perform a patient demographics query as PDQ specifies it. Whenever an attribute source resides in the patient domain, it should be ideally based on the PDQ supplier.

With regard to the subject domain, subject attributes (logical identifier of the principal and functional roles) are maintained by an identity provider and/or a dedicated attribute service. In order to issue an XUA assertion such services (e.g., an X-Assertion Provider) must get at these attributes. This could be achieved by carrying out a RFC 2798 (inetOrgPerson) or RFC 2256 (organizationalPerson) compliant query to a PWP directory. Whereas PWP is only for intra-enterprise use and provides personal data about physicians and employees, PDQ is inter-enterprise-enabled.

10 Appendix D: Glossary

The following terms are used in various places within this white paper, and are defined below. The definition of these is aimed at creating a common understanding of the core concepts of the white paper.

Access Control Policy

Set of rules to administer, manage, and control access to [network] resources [RFC 3060]

Access Control System (ACS)

The system entity that provides functionalities of access-control enforcement and decision.

Application Domain

Abstract domain referring to a clearly defined purpose of use that is realized by a software application/system. The domain holds attributes and policies of the application are located. The application domain is rather implicit because the semantics and the implied policies of an application are rarely expressed explicitly.

Brokered Trust

When one party trusts a second party who, in turn, trusts or vouches for, a third party (cp. WS Trust).

Circle of Trust (CoT)

A trust relationship among security token services that are able to exchange (authenticated) information across domain boundaries.

Context Domain

Refers to the location where attributes about the context (e.g., current step in a medical workflow) and local policies (e.g., Separation of Duty) are located. A service consumer is always acting inside this domain.

Direct Trust

When a relying party accepts as true all (or some subset of) the claims in the token sent by the requestor (cp. WS Trust).

Patient Domain

Refers to the location where attributes about the patient and the patients consent together with derived policies are located.

Policy Activation

A specific policy among a set of policies is selected and becomes “activated” for further transactions.

Policy Administration Point (PAP)

The system entity that creates a policy or policy set.

Policy Decision Point (PDP)

The point where policy decisions are made.

Policy Enforcement Point (PEP)

The point where policy decisions are actually enforced.

Policy Information Point (PIP)

The system entity that acts as a source of attribute values.

Policy Rendering

Refers to the process of evaluating a policy using existing attributes and formulating a policy decision.

Purpose of Use

Constraints the overall functionality of a dedicated healthcare system (that mediates or even initiates access to a protected resource) to the functionality needed within a certain medical treatment step or workflow.

Request Security Token (RST)

Message sent to a security token service in order to get issued a security token.

Request Security Token Response (RSTR)

Response message that contains the requested security token.

Resource

Especially in this context a resource refers to the location where medical information about patients is managed.

Resource Access Policy

A resource access policy controls who is able to access a protected resource within the context of a certain application. Resource access policies are enforces at each entry point to an application.

Resource Behavior Policy

A resource behavior policy defines how certain subjects might act on certain objects that are managed by a software application/system. (translation of purpose of use).

Resource Domain

Refers to the location where the protected resource is managed and attributes about the resource (metadata) as well as respective security policies are located.

Security Assertion Markup Language (SAML)

XML-based framework for exchanging authentication and authorization claims.

Security Token

A logical container for assertions made by security token services.

Security Token Service (STS)

A service that issues and validates security tokens.

Software Application/System

A computer program and/or a set of (distributed) software services that realize a clear purpose of use from a technical point of view.

Subject

Refers to a natural or juristic person identified by a distinct identifier.

Subject Domain

Refers to the location where attributes about a subject (roles etc.) are located. Subjects are identified and authenticated within this domain.

eXtensible Access Control Markup Language (XACML)

XML-based policy management specification and a policy framework.

eXtensible Rights Markup Language (XrML)

Language for specifying and managing access rights.

Web Services Trust (WS Trust)

A WS-* specification for e.g., issuing security tokens.

