SicAri – A security architecture and its tools for ubiquitous Internet usage

---

Deliverable

Protocols for Policy Negotiation

Work Package PE3

Version 0.1, 21. October 2005

Jan Peters, Fraunhofer IGD
Roland Rieke, Fraunhofer SIT
Taufiq Rochaeli, TUD-SEC
Bj"orn Steinemann, Fraunhofer SIT
Ruben Wolf, Fraunhofer SIT

Contents

1 Introduction

The scope of this work package in the context of the protocol engineering activities in the SicAri project is, to develop protocols for the negotiation and provisioning of policies in the SicAri architecture. This complements the research focused on policy patterns, the policy-refinement process, policy evaluation and policy specification languages in other SicAri work packages (see figure 1).

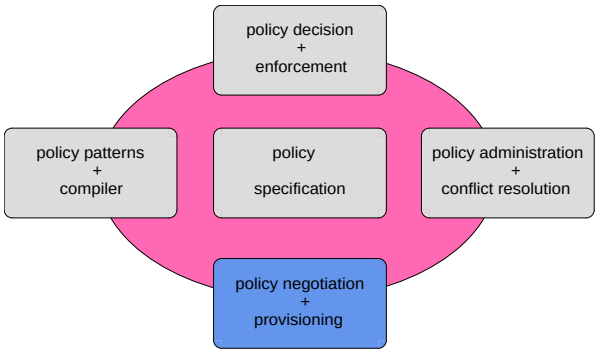

Figure 1: Policy related research in SicAri

Policies are a core concern in SicAri, particularly in the SicAri platform for the role based access rights management and enforcement. Another requirement for the developed protocol and usage concept is, that negotiation of global and local policies should be accomplished depending on the individual context and rule set.

To introduce some core entities and abbreviations used throughout this paper, figure 2 depicts a typical client/server architecture with a logical entity - the *Policy Decision Point (PDP)* - that makes policy decisions for itself or for other logical entities - the *Policy Enforcement Points (PEPs)* - that request such decisions.

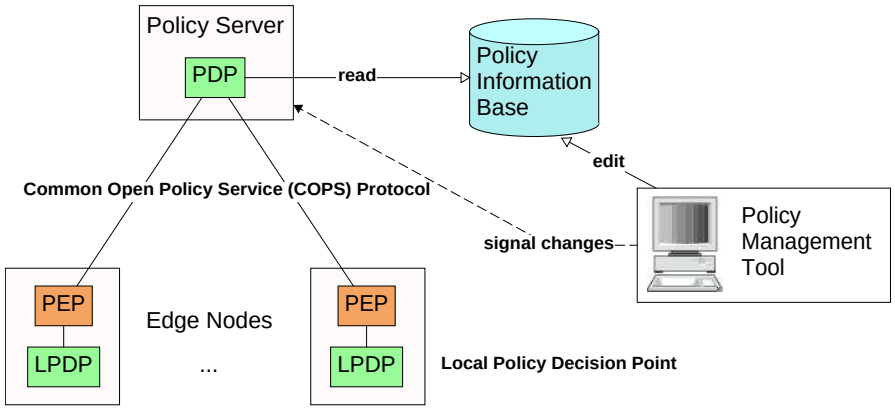

Figure 2: Generic architecture for policy control

The Policy Server gets the policy from the *Policy Information Base (PIB)*. This PIB in turn is

provided by the *Policy Management Tool (PMT)*.

In SicAri we have chosen the *Common Open Policy Service* (COPS) protocol [3] as protocol for policy distribution with the aim of providing an efficient and reliable means of provisioning multiple network devices. This decision is based on the work done in work package PE2 [18] where the requirements for a communication protocol between a policy server (PDP) and its clients (PEPs) to exchange and negotiate policy information as well as a specification of goals and preconditions and an evaluation of the suitability of some existing protocols and possible extensions are described in detail. Main advantages of COPS are the effective decoupling of the protocol from the information model and its support for two common usage models for policy control, namely the outsourcing mode and the provisioning (or configuration) mode.

Section 2 covers the latest research in the field of policy negotiation and the Web Service Policy Language (WSPL).

Section 3 describes some policy negotiating scenarios and covers the bootstrapping, policy provisioning and service access in the SicAri platform as well as derived requirements.

Section 4 covers the base technologies used throughout the SicAri framework such as the Common Open Policy Service (COPS) protocol and extensions thereof, the main concepts of Role-Based Access Control (RBAC) and the Extensible Access Control Markup Language (XACML).

Section 5 describes the sicari policy framework with a focus on architecture and integration of the policy negotiation and provisioning protocol and the logical entities such as policy enforcement points (PEPs) and policy decision point (PDP) that are using this facility.

2 State of the Art

This section covers the latest research in the field of policy negotiation.

2.1 Web Service Policy Language

Web Service Policy Language (WSPL) is aiming to provide the mechanisms needed to enable Web services applications to specify policy information, which contains the security requirements and security capabilities of web services applications. It is designed only to specify the security policies in order to establish the connection between two endpoints of web service application scenario. Thus, this standard does not support more complex, application-specific policies that *take effect after the connection* is established. In order to establish the interoperability between the web services application, the policy is expressed in a XML Infoset representation. It has the XML flavour in its syntax. The WSPL, as specified in the specification, has the following constructions: **policy assertions**, **policy alternatives** and **policy**.

2.1.1 Elements of WSPL

A(n) (policy) assertion is the basic and atomic unit of a policy. As an example, an assertion could declare that the message should be signed. It should be noted that the defined terms and their interpretation *are domain specific*, and *not defined* in WSPL specification. An assertion is defined by a unique Qualified Name, and can be a simple string or a complex object with many sub elements and attributes. The assertion is also extensible, which can be customised to express the policies for quality of service, privacy, security, etc. We will outline the general constructs of WSPL that enable the policy negotiation.

A(n) (policy) alternative is a collection of assertions, which may be an empty collection. In this case, the policy thus has no behaviours. The vocabulary of an alternative must be a set of all assertion types used in the policy of corresponding domain.

A policy is a collection of alternative, which may be an empty collection. In case of a policy having no alternative, it has no choice regarding the security requirements (or capabilities).

The alternatives and the assertions are combined by using the operators `wsp:All` and `wsp:ExaclyOne`, in order to follow the normative form. Along with the operators, these constructs play an essential role in policy negotiation, which occurs before the connection establishment.

2.1.2 WSPL Operators

The operators are intended to express relationships between policy assertions or policy alternatives, treating the opaquely.

The operator `wsp:All` is interpreted as: all the assertions should be satisfied in order to evaluate this expression into true.

According to [10], the operator `wsp:ExactlyOne` has two possible interpretations:

- one or more of the alternatives should be satisfied in order to evaluate this expression into true. In this case, the `wsp:ExactlyOne` is an inclusive OR.

- *exactly* one alternative should be satisfied in order to evaluate this expressions into true.

These operators have the commutative, associative and distributive laws between two of them. A more comprehensive description of these laws can be found in the WSPL standard [9].

2.1.3 Interpreting the WSPL Elements

An assertion is supported by a WS client if and only if the client satisfies the requirement (or accommodates the capability) corresponding to the assertion. An alternative is supported by a client if and only if the client supports all the assertions in the alternative. A policy is supported by a client if and only if the client supports at least one of the alternatives in the policy.

2.1.4 Policy Negotiation

The policy negotiation of WSPL is handled in WSPL standard by using *Policy Intersection* function. Policy intersection aims at finding the mutually compatible policy alternatives, which are expressed by two or more parties.

The function takes two policies as its parameters, and returns a policy. This function is associative and commutative.

It should be noted that the assertions may contain domain-specific extension, a domain-specific policy processing may be required.

In general, the intersection function searches for alternatives having the same vocabulary [1] from two policy instances, and collects the mutual alternatives into a single common policy instance. The collected mutual alternatives should be the subset of alternatives of both initial policy instances.

2.1.5 Conclusion

The first version of WSPL standard was released in 2003, which is more extended than the recent version of WSPL. For example, the policy operator `ws:OneOrMore` was introduced in the first version, but omitted in the second version of WSPL with the consequence extending interpretation of the policy operator `ws:ExactlyOne`.

The WSPL Policy only defines the policies which are needed prior to connection establishment. It does not express the policies, which are enforced during the connection establishment.

The standard WSPL Policy is kept general and extensible, and intended to cope more specific policies in certain domains, such as security, privacy, etc.

[1] In this case, it considers only policy expression from the same domain.

3 Policy Negotiating Scenarios

3.1 Bootstrapping

Distributed systems with policy based network and security management benefit from the fact that the policy data can be alloted from a central point of administration. This being an advantage for most of the time there are still some subtle questions that have to be respected. One of these issues is the phase of bootstrapping the system.

While the operation of the running system usually is not problematic the bootstrapping needs some extra observance.

- Within SicAri, every single platform instance is started by the local administrator, who has to authenticate against the platform during the bootstrappinbg process. As consequence all basic services are started, initialized, and monitored on behalf of this platform administrator. Platform administrators are a special kind users registered by the infrastructure-global identity manager.
- After the platform instance has been started successfully, a globally registered user is able to log into the plattform to subsequently access available services and resources. The user's access rights are defined by a global security policy.
- With respect to access control and when neglecting role-based partitioning of permissions among a bunch of administrators, a local administrator as single root can be handled in two different way. Either, all permissions on local resources are granted according to a local security policy and a global security policy defines additional rules when accessing remote resources. Or, the global security policy even defines rules to restrict the local administrator on the local platform.
- In the first case, bootstrapping can be done according to a local policy for the administrator. Generally, this should not be a security problem for the platform infrastructure, since the locally used services are provided by some local user (i.e. the local administrator?! - cf. comments below).
- The second case is based on a consistent security policy concept with only one globally defined security policy for all platforms in the infrastructure. The bootstrapping of a platform can only be done autonomously (i.e. without network connection), if the LPDP caches a valid (e.g. according to a timeout) policy in a persistent manner, which can be used.
- In either case, basic services or service stubs try to connect to remote services during the bootstrapping process (e.g. if LPDP tries to update an outdated security policy). The role 'platform adminitrator' can encapsulate the special permissions for this interaction (to be enforced on the remote platforms). Thus, basic services have to be started after the platform administrator's authenticiton process, and thereby already within the security context of a valid SicAri user.

3.2 Policy Provisioning

Policy based control of networks and computer systems has the benefit that the control part of the system is kept decoupled from the rule base for the governing decisions. This enables the administrators to easily run, manage and change the systems' behaviour without having to modify the software or the controlled nodes. The system is controlled by policies that specify behaviour rules which are interpreted by decision components and are asserted by enforcement components. Hence one just adapts the policy rules to eventually changed conditions usually using a central administration component. The changes are then spread out to the participating nodes through special provisiong protocols.

When new components or nodes join the network they have to be equiped with the relevant policy. An initial configuration or service location protocol informs the component about the location of the server that provides the policies. The components then contact the policy server and request their initial policy. This way all components receive the rule base for control decisions right after bootup without the need to have an administrator to manually configure them. The MSC 3 delineates this process.

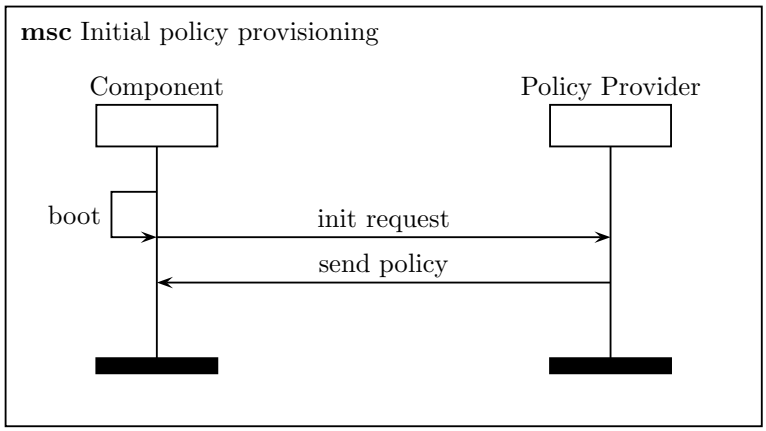

Figure 3: Initial policy provisioning

3.3 Service Access

In this section we want to understand a policy only in the terms of security policy. In Subsetion 3.2 we looked at the case that a policy server provided security policies for the different nodes and components in the network. This assumes that the concerned components are able to interprete these policies and can derive own decisions from the policy rules. This means that the component implements a (Local) Policy Decision Point (L)PDP.

A different possible scenario would be a component or network node that may not have the computational power to infer decisions from a given security policy or may not have enough memory to hold the complete policy and the decision logic. In this case it would outsource the decision finding to a server that offers a Policy Decision Point (PDP). Whenever this component is requested access to one of it's resources or to local data it asks the PDP whether the access

should be permitted or denied. To improve performance and save bandwidth of the network the component would store the PDP's decisions and use them when the same access requests are repeatedly issued.

Picture 4 depicts these two scenarios very abstractly. The Policy Enforcement Point (PEP) contacts either the PDP server with the master policy for a decision or relies on the Local PDP to do this.

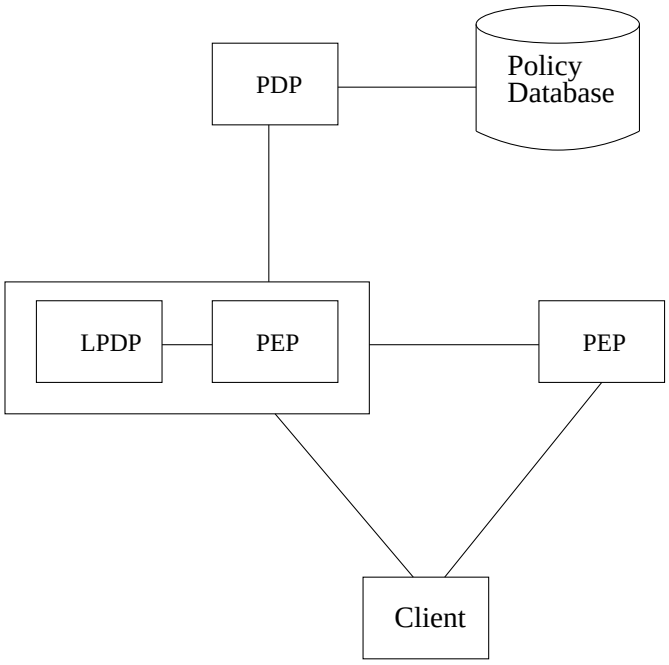

Figure 4: $SicAri$ Architecture for policy provisioning

In both scenarios — decisions are made locally or are outsourced — the component has to be able to react to the case when the connection to the policy master server is broken. Keep in mind that the strength of the policy based security and network management is the approach to centrally administer the rule base for the policy. This demands the timely distribution of policy changes and updates to the participating components and bears the risk that a remote node might rely on outdated information when the network connection is disrupted for a longer period of time. Therefore precautions have to be taken to minimize the risk of making wrong decisions. The next two subsections will discuss this challenge.

PDP Connection Alive

A protocol that controls the communication between a policy managed component and it's policy server would require them to exchange keep-alive messages from time to time. This way both parties assure that they have a stable connection and the counterpart is alive. The policy managed component knows that it can reyly on it's locally stored policy or on it's saved copies of the PDP's decisions.

The MSC 5 depicts the scenario of the left component from picture 4 which makes use of the LPDP to answer the access request from the client. Whenever the corresponding security policy is changed the PDP unsolicitedly sends an update message to the PEP which in turn may now

answer the client's access requests differently than before. As long as the PEP receives keep-alive messages from the PDP the component knows that the LPDP's policy base is still valid.

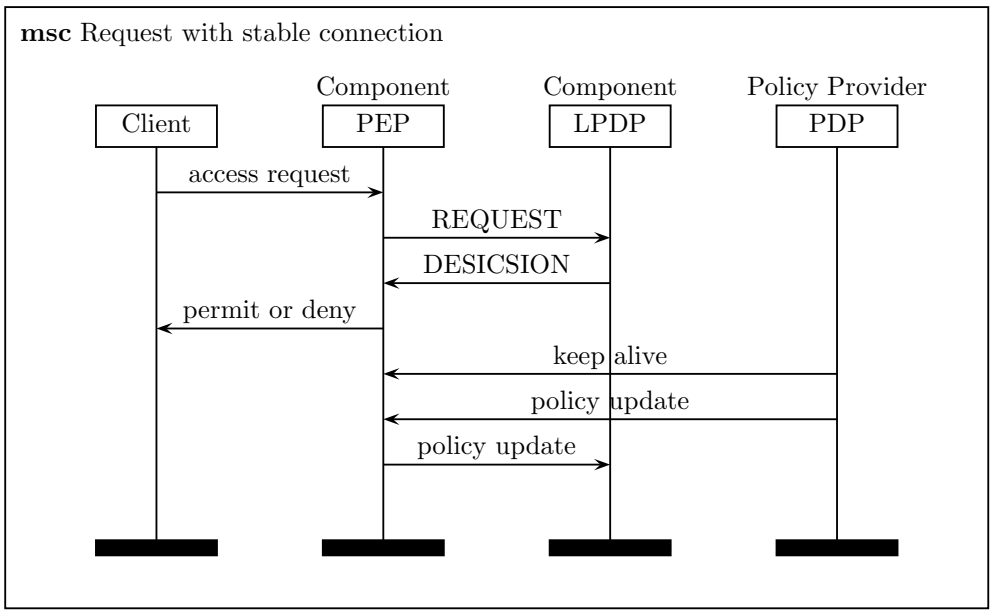

Figure 5: Request with stable connection

The scenario for the component that outsources the decision process to the PDP looks very similar. There is no need for update messages from the PDP to the PEP since the component does not hold a copy of the policy. If the solution for this scenario allows the PEP to cache decisions from the PDP it may use this information to locally answer identical or similar requests. Since these decisions may not be vaild for the lifetime of the system we have to consider a timeout. Either the validity timer for the decision can not be reset and runs out after a fixed period of time or it can be reset whenever a new message from the PDP arrives at the PEP. The first case gets by without keep-alive messages but will delete the decions earlier then the second solution. In the latter case we need a keep-alive message and the PDP to remember the decisions that it made. Whenever the policy changes the PDP has to explicitly revoke those decisions from the PEP for which the inference basis has been affected. This solution is depicted in 6.

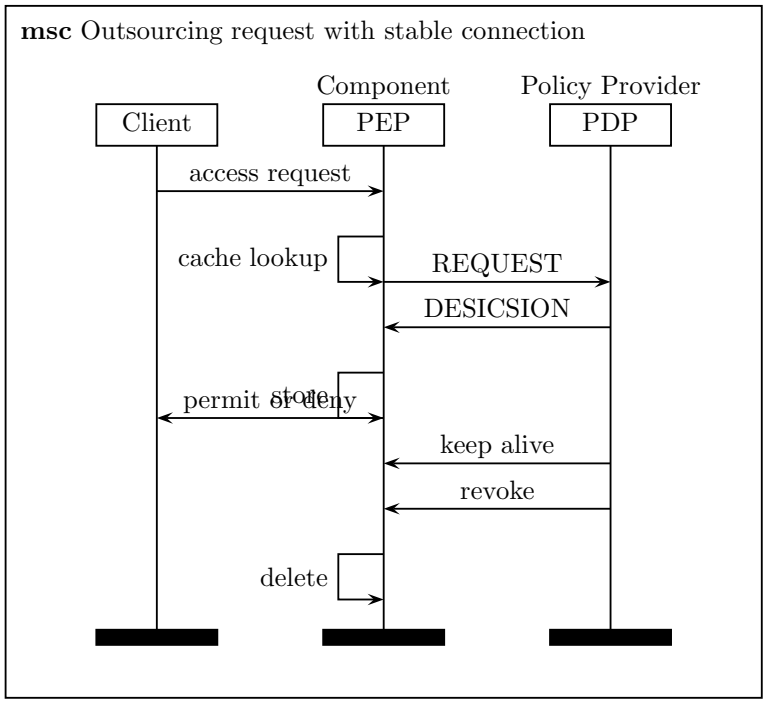

Figure 6: Outsourcing request with stable connection

PDP Connection Interrupted

When network problems occur or when the PDP is simply not available the PEPs have to have a fallback solution at their disposal. When the application that is controlled by the security policy can not achieve up to date decisions from the PDP or when it's local policy becomes invalid, it still has to protect the data and the resources that it is holding in trust. For this case a meta security policy has to be at hand that makes sure that no data is compromised or accessed by unauthorized parties. Depending on the criticality of the application this might even mean that no access is granted at all.

3.4 Derived Requirements

The previous three subsections introduced the major questions that a policy negotiation and provisioning protocol has to be able to answer. From these scenarios the following requirements can be derived:

4 Technologies

4.1 The COPS (Common Open Policy Service) Protocol

RFC 2748 [3] defines the COPS protocol, a simple query and response protocol that can be used to exchange policy information between a policy server (PDP) and its clients (PEPs).

Figure 7 depicts the protocol elements and their attribution to the logical entities PDP and PEP.

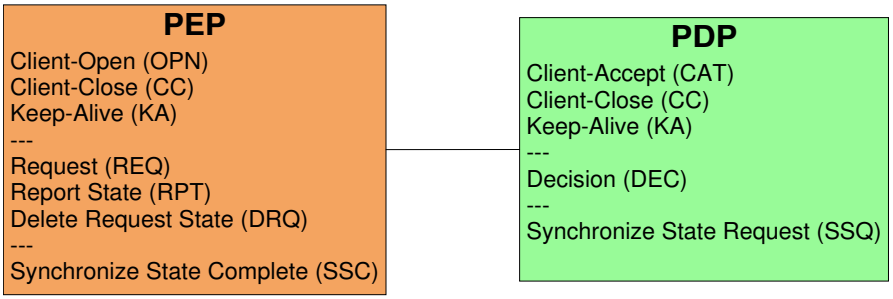

Figure 7: COPS protocol elements

COPS effectively decouples the protocol from the information model.

COPS supports two common models for policy control, namely the outsourcing mode and the provisioning (or configuration) mode.

4.1.1 The COPS outsourcing mode

In the outsourcing scenario, when a PEP receives an event that requires a policy decision, it creates a request, which includes information that describes the admission-control request.

1. The PEP may consult the local configuration database to identify which policy elements can be evaluated locally, passes the request to the local policy decision point (LPDP) and acquires the results.
2. The PEP then passes all the policy elements and the partial result to the PDP.
3. The PDP combines its result with the partial results from the LPDP and returns the final policy decision to the PEP.
4. Since the request is stateful, the request will be remembered, or installed, on the remote PDP. The unique handle, specified in the request identifies this request state.
5. The PEP can update a previously installed request state by reissuing a request for the previously installed handle. The remote PDP is then expected to make new decisions and send a decision message back to the PEP.
6. In addition, outsourcing decisions from the PDP MAY result in a corresponding solicited Report State from the PEP depending on the context and the type of client.

7. Likewise, the server MAY change a previously issued decision on any currently installed request state at any time by issuing an unsolicited decision message. At all times the PEP module is expected to abide by the PDP's decisions and notify the PDP of any state changes.

8. The PEP is responsible for deleting the request state once the request is no longer applicable.

The Report State message may also be used to provide periodic updates of client specific information for accounting and state monitoring purposes depending on the type of the client.

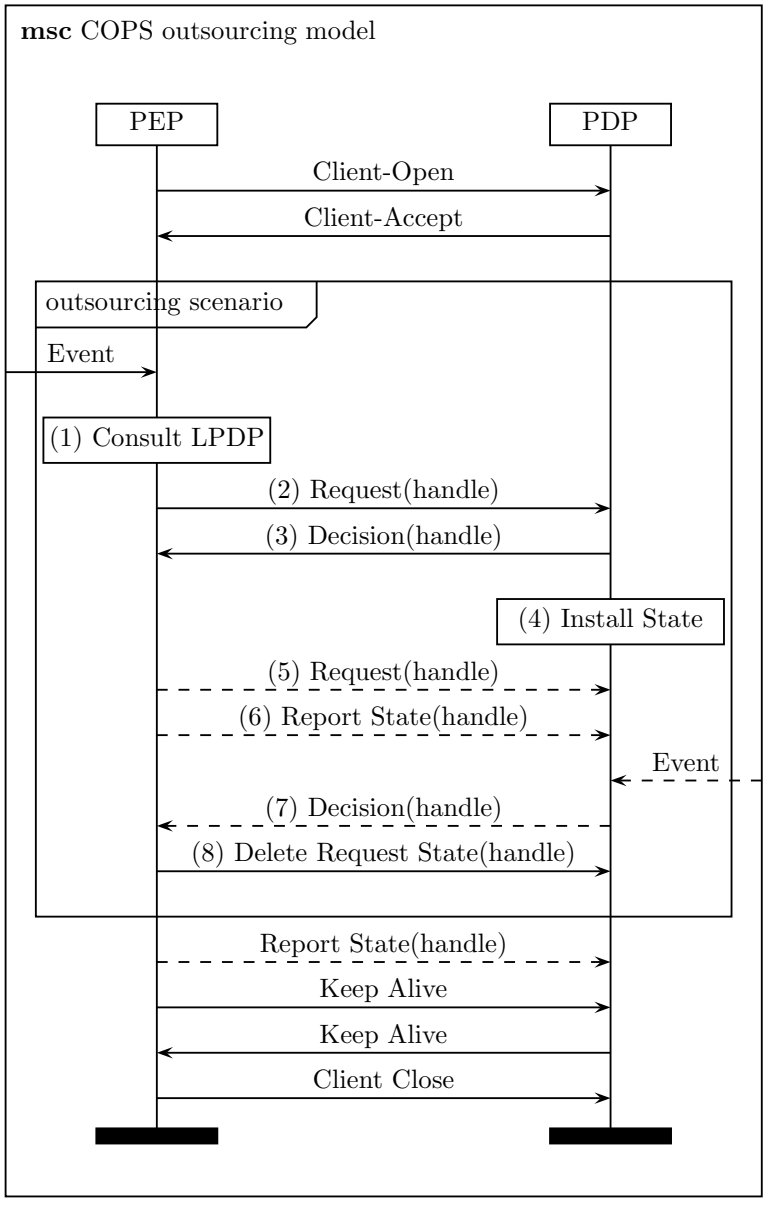

Figure 8: COPS outsourcing mode

RFC 2749 [8] describes how the COPS protocol is used to provide for the outsourcing of policy decisions for the Resource reSerVation Protocol (RSVP).

4.1.2 The COPS provisioning mode

The PDP provisions the configurations to the PEPs and when the PEP receives a request it processes it without interacting with the PDP since policies has already been enforced.

1. At first the PEP will make a configuration request to the PDP for a particular interface, module, or functionality that may be specified in the named client specific information object.
2. The PDP will then send potentially several decisions containing named units of configuration data to the PEP.
3. The PEP is expected to install the configuration locally.
4. The PEP MAY notify the remote PDP of the local status of an installed state using the report message where appropriate. The report message is to be used to signify when billing can begin, what actions were taken, or to produce periodic updates for monitoring and accounting purposes depending on the client. This message can carry client specific information when needed.
5. The PEP now uses the configuration locally to process events without further need to contact the PDP.
6. A particular named configuration can be updated by the PDP by simply sending additional decision messages for the same named configuration. The same mechanism can be used by the PDP to send a decision flags object with the remove configuration command.
7. The PEP SHOULD then proceed to update/remove the corresponding configuration.
8. The PEP SHOULD send a report message to the PDP that specifies that the local configuration has been modified/deleted.

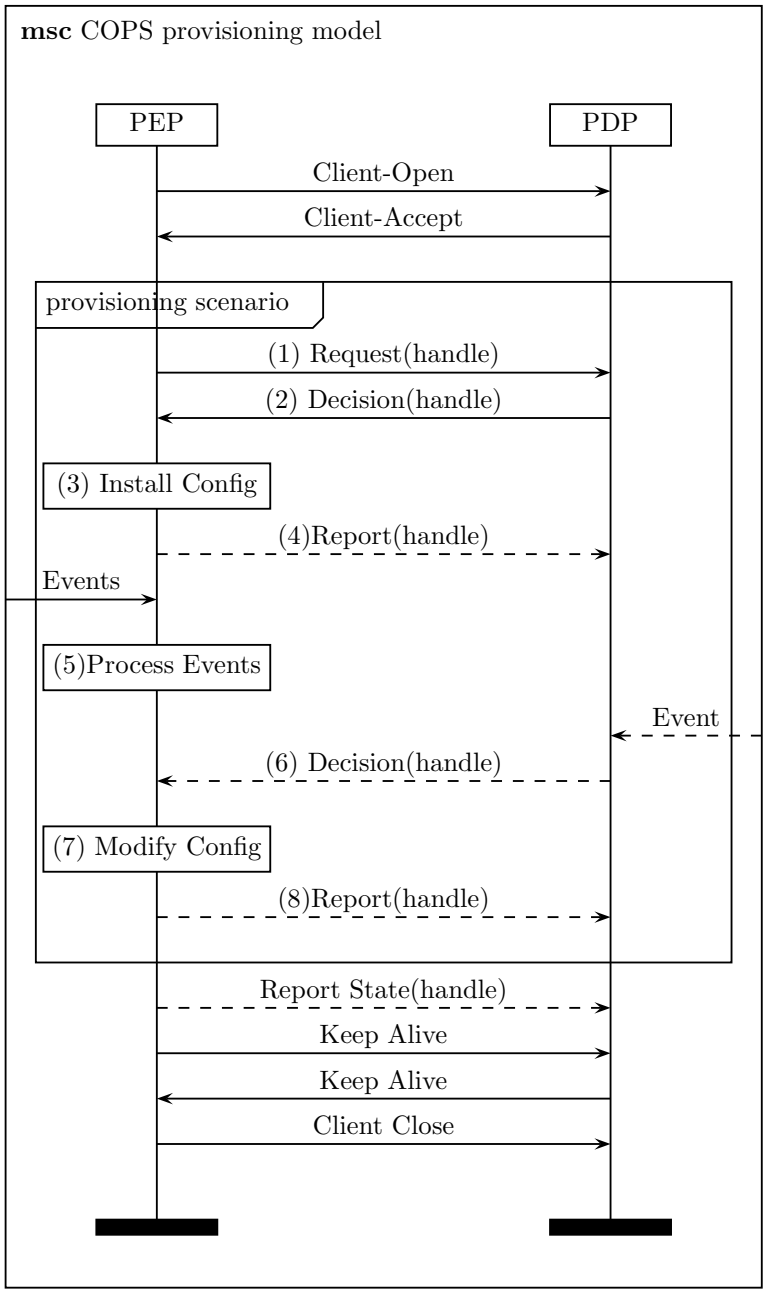

Figure 9: COPS provisioning mode

An example for the usage of the COPS protocol, for the provisioning of policy, is introduced in RFC 3084 [1]. In this provisioning model, the policy information is viewed as a collection of Provisioning Classes (PRCs) and Provisioning Instances (PRIs) residing in a virtual information store, termed the Policy Information Base (PIB). Collections of related Provisioning Classes are defined in a PIB module.

The COPS-PR protocol [1] describes the use of the COPS protocol for support of policy provisioning. This specification is independent of the type of policy being provisioned (QoS, Security, etc.) but focuses on the mechanisms and conventions used to communicate provisioned information between PDPs and PEPs. The protocol extensions described in this document do not make any assumptions about the policy data model being communicated, but describe the message formats and objects that carry the modeled policy data.

In COPS-PR, policy requests describe the PEP and its configurable parameters (rather than an operational event). If a change occurs in these basic parameters, an updated request is sent. Hence, requests are issued quite infrequently. Decisions are not necessarily mapped directly to requests, and are issued mostly when the PDP responds to external events or PDP events (policy/SLA updates) such as notification about changes in the PIB.

The data carried by COPS-PR is a set of policy data. The protocol assumes a named data structure (PIB), to identify the type and purpose of unsolicited policy information that is "pushed" from the PDP to the PEP for provisioning policy or sent to the PDP from the PEP as a notification.

RFC 3159 [11] describes the Structure of Policy Provisioning Information (SPPI). Secifically it defines the adapted subset of SNMP's Structure of Management Information (SMI) used to write Policy Information Base (PIB) modules.

COPS extensions can be written by specification of special PIBs (Policy Information Bases) using the SPPI language.

RFC 3318 [20] defines a set of PRovisioning Classes (PRCs) and textual conventions that are common to all clients that provision policy using the COPS protocol for provisioning.

4.1.3 COPS-MU (mobile user) and COPS-MT (mobile terminal)

Originally the IETF policy-based networking is defined for application to network nodes. Recent work as for example [2] suggest to extending the policy-based networking to the end-user terminals (home terminal,PDA,mobile phone,...). This is based on variants of the COPS architecture and usage patterns and the exchanged policy information to support such new policy-aware terminals.

In this proposed framework, terminals are policy aware. COPS-MU supports mobile IP registration based on policy controled mechanisms. The Terminal Policy Enforcement Point (TPEP) interacts with the policy decision point using COPS. Terminal PEPs in this framework can be black-boxes or smart-card based.

COPS-MT in this context extends COPS to support location management.

In the proposed COPS-MT framework, the foreign PDP (FPDP) behaves as a proxy for the mobile terminal to forward its registration requests. It behaves as a server to the terminal PEP (TPEP) and as a client to the home PDP (HPDP) using COPS. The registration procedure achieved by the terminal PEP in COPS-MT using a home agent (HA) as in mobile IP. The HA uses its PEP to update the registration requested by the mobile terminal. Figure 10 depicts a typical interaction in the COPS-MT framework (similar interaction schemes with different content such as registration or authentication information are proposed in this research work).

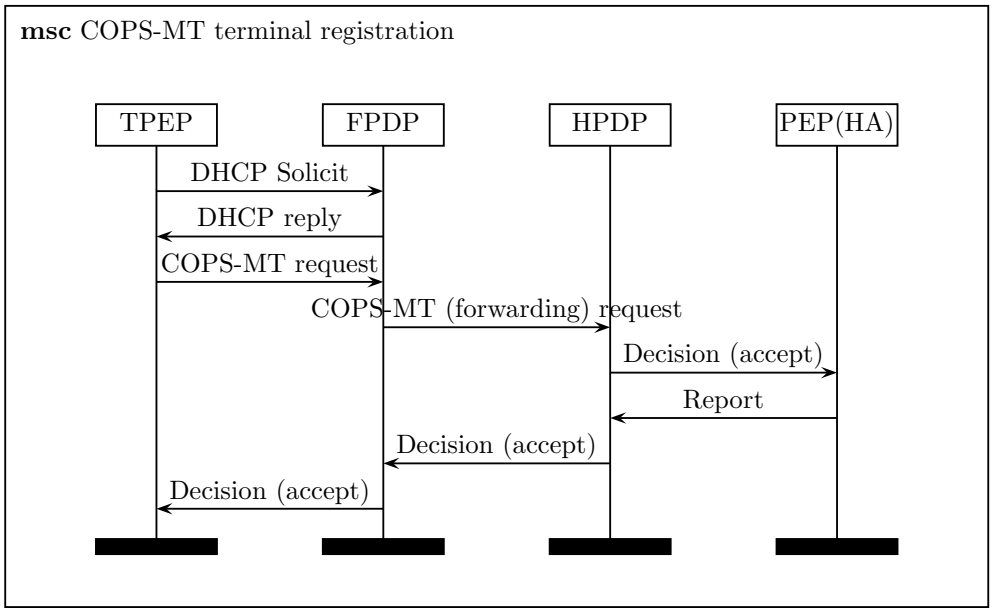

Figure 10: COPS-MT terminal registration

4.1.4 COPS inter-domain policy negotiation

In his dissertation "Policy-Based Quality of Service Management in Wireless Ad Hoc Networks" [17] K. Phanse proposes to extend COPS for a scenario with a visiting client (PEP_H) in a foreign domain. The PDP of the foreign domain (PDP_F) negotiates with the home PDP (PDP_H) of the client if there is no local policy available for the client. Details of this usage of the COPS protocol have already been described in work package PE2 [18].

4.2 Role-Based Access Control (RBAC)

Since the general concepts of RBAC are well-understood and extensively described in the literature [5, 21, 4, 12], this section only presents the main concepts of RBAC.

Basic idea. The central terms in RBAC are *user*, *role*, and *permission* (see Figure 11). Therefore, we have sets *USERS*, *ROLES*, and *PERMS*. For elements of these sets, we have two assignment relations $UA \subseteq USERS \times ROLES$ and $PA \subseteq PERMS \times ROLES$. The *user assignment UA* defines a relation between users and roles, whereas the *permission assignment PA* defines the relation between roles and permissions. Both relations are many-to-many. The set of all permissions is obtained by the combination of *operations* and *objects*, i.e., $PERMS = OPS \times OBS$, where *OPS* and *OBS* are the corresponding sets. Types of operations depend on the type of system which is considered. In access control terminology, an object is an entity which contains or receives information, e.g., an object may be a file or some exhaustible system resources. If a user $u \in USERS$ changes to a new user category leaving his old role r_{old} then he is simply assigned to a new role r_{new} by getting the permissions of r_{new} and losing the permissions of r_{old}.

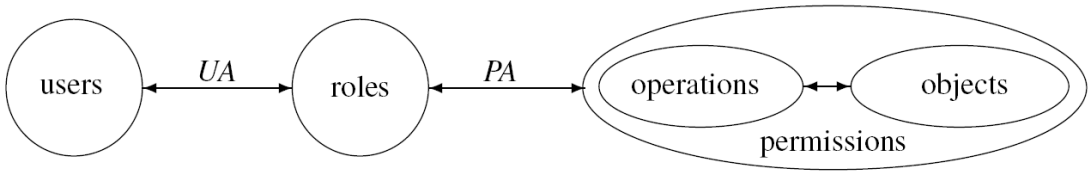

Figure 11: Core RBAC model

Role hierarchies are thought to be one of the most desirable features in RBAC. They are very useful when overlapping capabilities of different roles result in common permissions because they allow to avoid repeated permission-role assignments. This allows to gain efficiency, e.g., when a large number of users is authorized for some general permissions. Role hierarchies $RH \subseteq ROLES \times ROLES$ are constructed via inheritance relations between roles, i.e., by the introduction of senior and junior relations between roles $r_1 \succeq r_2$ in such a way that the senior role r_1 inherits permissions of the junior role r_2.

Role activation and the principle of least privilege. The principle of least privilege requires that a user should not obtain more permissions than actually necessary to perform his task, i.e., the user may have different permissions at different times depending on the roles which are actually activated. As a consequence, permissions are revoked at the end of sessions.

Further relevant properties. RBAC allows the specification of *separation of duty* statements within policies in order to cover conflict of interest situations. RBAC is assumed to be policy-neutral. This means that RBAC provides a flexible means to deal with arbitrary security policies. It is highly desirable to have a common means to express a huge range of security policies. RBAC facilitates security management, makes it more efficient, and reduces costs.

Within SicAri, access control polcies are based on the RBAC standard. Implementation of role-based access control policies is done by means of the Extensible Access Control Markup Language (XACML).

5 SicAri Policy Framework

5.1 Policy negotiation between PDP and PEP

A usage scenario of COPS for the adjustment of two policies has been suggested in [18]. Figure 12 illustrates this scenario.

This scenario "abuses" the report state (RPT) message that is normally used by the PEP to communicate to the PDP its success or failure in carrying out the PDP's decision, or to report an accounting related change in state. In this proposal the report of a failure means the denial of the proposed policy by the PEP.

By designing a specific PIB (Policy Information Base), no change is needed to the COPS protocol itself. In this way the SicAri framework can build on the existing COPS technology without having to revisit the protocol every time new kinds of policy information needs to be carried by the protocol.

As depicted in Figure 12, the policy client negotiates with the policy server using the COPS protocol. This negotiation process can take place within an administrative domain or between administrative domains.

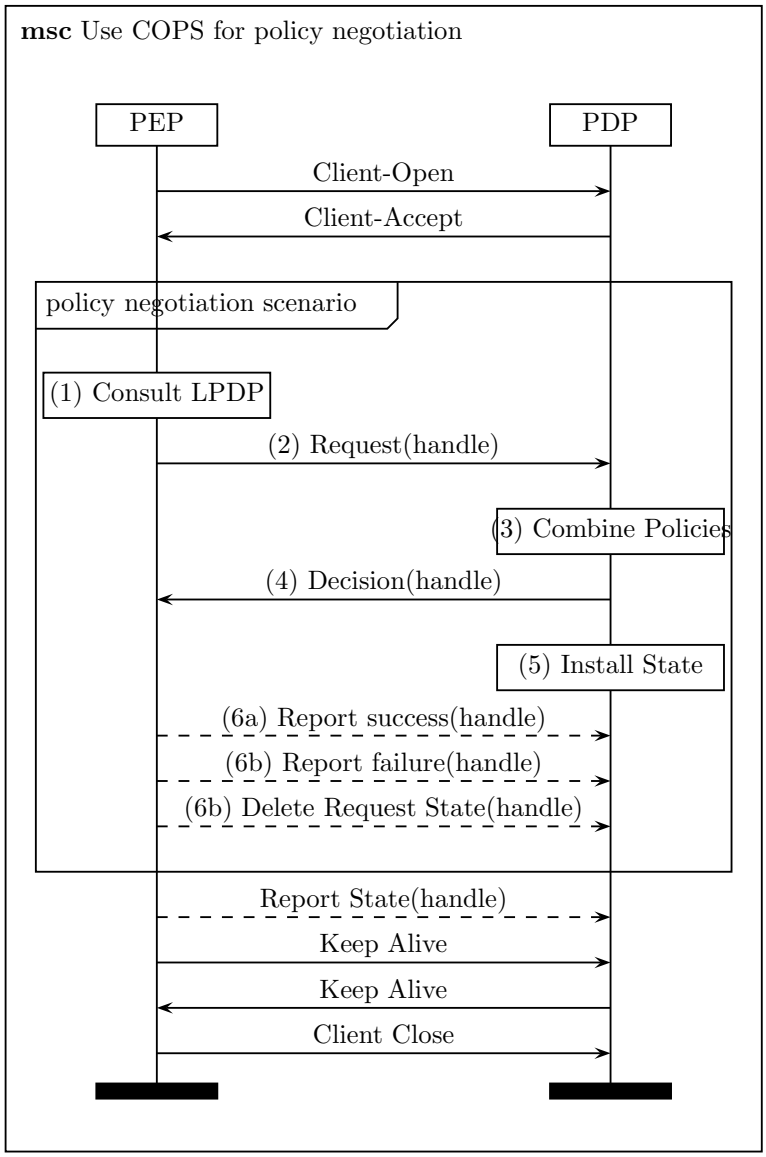

Figure 12: COPS policy negotiation

1. The PEP consults the local configuration database to identify which policy elements can be evaluated locally, passes the request to the local policy decision point (LPDP) and acquires the results.

2. The PEP then passes all the policy elements and the partial result to the PDP.

3. The PDP combines its result with the (partial) results from the LPDP. How to handle possible rule conflict resolution is out of scope of the protocol.

4. The resulting policy is send to the PEP as decision.

5. Since the request is stateful, the request will be remembered, or installed, on the PDP.

6. (a) The PEP now can accept this decision and communicate to the PDP its **success** in carrying out the PDP's decision using the *Report State (RPT)* protocol element.

   (b) The PEP can alternatively deny the PDPs decision and report this as a **failure** to implement the PDP's decision locally.

   In case of **failure** the PEP sends a *Delete Request State (DRQ)* element to reset the state of the PDP and then probably starts negotiation with a new policy proposal.

The rules for conflict resolution between different policies of the PEP and the PDP are dependend on the context and on the type of policies to be negotiated. The COPS protocol is independent of policy content and so it supports only negotiation elements but not conflict resolution procedures.

References

[1] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. COPS Usage for Policy Provisioning (COPS-PR). RFC 3084, IETF, March 2001.

[2] Hakima Chaouchi. A new policy-aware terminal for qos, aaa and mobility management. *Int. J. Netw. Manag.*, 14(2):77–87, 2004.

[3] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748, IETF, January 2000.

[4] David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli. *Role-Based Access Control*. Computer Security Series. Artech House, Boston, 2003.

[5] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, August 2001. URL: http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.

[6] Fraunhofer Institute for Secure Telecooperation SIT, Darmstadt. *Simple Homomorphism Verification Tool – Manual*, 2002.

[7] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, IETF, December 2002.

[8] S. Herzog, Ed., J. Boyle, R. Cohen, D. Durham, R. Rajan, and A. Sastry. COPS usage for RSVP. RFC 2749, IETF, January 2000.

[9] IBM, BEA Systems, Microsoft, SAP AG, Sonic Software, and VeriSign. Web services policy framework. URL: ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf

[10] Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler. Representing web service policies in owl-dl. In *LNCS*, 2005 (to appear).

[11] K. McCloghrie, M. Fine, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer. Structure of Policy Provisioning Information (SPPI). RFC 3159, IETF, August 2001.

[12] National Institute of Standards and Technology (NIST). Role-Based Access Control. URL: http://csrc.nist.gov/rbac/.

[13] P. Ochsenschl ger, J. Repp, and R. Rieke. Abstraction and composition – a verification method for co-operating systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 12:447–459, June 2000. Copyright: ©2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[14] P. Ochsenschl ger, J. Repp, and R. Rieke. The SH-Verification Tool. In *Proc. 13th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2000)*, pages 18–22, Orlando, FL, USA, May 2000. AAAI Press. Copyright: ©2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[15] P. Ochsenschl ger, J. Repp, and R. Rieke. *Simple Homomorphism Verification Tool – Tutorial*. Fraunhofer Institute for Secure Telecooperation SIT, Darmstadt, 2002.

[16] P. Ochsenschl ger, J. Repp, R. Rieke, and U. Nitsche. The SH-Verification Tool – Abstraction-Based Verification of Co-operating Systems. *Formal Aspects of Computing, The International Journal of Formal Method*, 11:1–24, 1999.

[17] Kaustubh S. Phanse. Policy-based quality of service management in wireless ad hoc networks. Dissertation, Virginia Polytechnic Institute and State University, August 2003.

[18] Roland Rieke and Taufiq Rochaeli. Bericht zum Arbeitspaket PE2 im Projekt SicAri "Formale Spezifikation von Zielen und Voraussetzungen". Technical report, Fraunhofer - Institute Secure Telecooperation, 2004.

[19] Roland Rieke and Carsten Rudolph. Bericht zum Arbeitspaket PE1 im Projekt SicAri "Anforderungsanalyse". Technical report, Fraunhofer - Institute Secure Telecooperation, 2004.

[20] R. Sahita, Ed., S. Hahn, K. Chan, and K. McCloghrie. Framework Policy Information Base. RFC 3318, IETF, March 2003.

[21] Ravi Sandhu. Role activation hierarchies. In *Proceedings of the third ACM workshop on Role-based access control*. ACM Press, 1998.

[22] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for Policy-Based Management. RFC 3198, IETF, November 2001.

6 Appendix

6.1 Terminology and Abbreviations

AAA - Authentication, Authorization, Accounting

AD - Administrative Domain

APA - Asynchronous Product Automata [13, 19]

CIM - Common Information Model

COPS - Common Open Policy Service Protokoll [3]

COPS-PR - COPS Policy Provisioning Protokoll [1]

IETF - Internet Engineering Task Force

LPDP - Local Policy Decision Point
An optional Local Policy Decision Point can be used by a device to make local policy decisions in the absence of a PDP

MIB - Management Information Base

PBNM - Policy-Based Network Management

PDP - Policy Decision Point
A logical entity that makes policy decisions for itself or for other network elements that request such decisions

PEP - Policy Enforcement Point
A logical entity that enforces policy decisions

PIN - Policy Ignorant Node
A network component that does not support policy control

PIB - Policy Information Base

PKI - Public Key Infrastructure

PMT - Policy Management Tool

Policy
Rules that define criteria for administration, acccess and use of resources

Policy control
Application of policy rules to control access to resources

QoS - Quality of Service

RSVP - Resource reSerVation Protocol

SHVT - SH Verification Tool [6, 15, 14, 16]

SLA - Service Level Agreement

SLS - Service Level Specification

SMI - Structure of Management Information

SNMP - Simple Network Management Protokoll [7]

SPPI - Structure of Policy Provisioning Information [11]

For further terminology and abbreviations see also rfc3198 [22] “Terminology for Policy-Based Management”.