Policy Evaluation, Enforcement and Management Architecture

Approved Version 1.0 – 24 Jul 2012

Open Mobile Alliance

OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-20120724-A

Contents

Figures

Tables

1. Scope (Informative)

This document provides the architecture for the Policy Evaluation, Enforcement and Management (PEEM) enabler of OMA.

The PEEM enabler evaluates and/or enforces policies. Policies are applied to requests to, or responses from resources or, when explicitly called by a resource.

The architecture shown in this document is intended to facilitate the development of specifications for defining, managing, evaluating, and enforcing policies in a way that is scalable and flexible yet independent of any specific implementation scheme. Additionally, the architecture enables reuse by other enablers so that their requirements are satisfied.

Note also that this enabler does not specify individual policies, but rather addresses requirements on how to express policies.

2. References

2.1 Normative References

[PEEM_RD] "Policy Evaluation, Enforcement and Management Requirements", Version 1.0 Open Mobile Alliance™, OMA-RD_Policy_Evaluation_Enforcement_Management-V1_0, URL: http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL: http://www.ietf.org/rfc/rfc2119.txt

2.2 Informative References

[3GPP-OSA-policy-mgmt] 3GPP TS 29.198-13, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Open Service Access (OSA); Application Programming Interface (API);Part 13: Policy management Service Capability Feature (SCF) (Release 6)", URL: http://www.3gpp.org/ftp/Specs/html-info/29198-13.htm

[3GPP TS 22.071] "Location Services (LCS); Service description; Stage 1 (Release 7)", URL: http://www.3gpp.org/ftp/Specs/latest/Rel-7/22_series/22071-720.zip

[3GPP TS 23.125] **"**Overall high level functionality and architecture impacts of flow based charging; Stage 2 (Release 6)", URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.125/23125-650.zip

[BPEL] "Business Process Expression Language", OASIS, URL: http://www.oasis-open.org/specs/index.php#wsbpelv2.0

[BPML] "Business Process Modeling Language", URL: http://www.bpmi.org/bpml-spec.htm

[COMMONPOL] "A Document Format for Expressing Privacy Preferences", H. Schulzrinne, J. Morris, H. Tschofenig, J. Cuellar, J. Polk, and J. Rosenberg, February 2007, URL: http://www.ietf.org/rfc/rfc4745.txt?number=4745

[FTPrfcs] "FTP Protocol Related Documents", URL: http://www.wu-ftpd.org/rfc/

[ISO/IEC 9899] ISO/IEC 9899, "Programming languages – C", URL: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n869/n869.txt.gz

[J2SEBLOB] "Interface Blob", java.sql, J2SE v.1.4.2, URL: http://java.sun.com/j2se/1.4.2/docs/api/java/sql/Blob.html

[Java] "The Java Language Specification", Sun Microsystems, URL: http://java.sun.com/docs/books/jls/

[OMA-DICT] "OMA Dictionary", Version 2.6, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_6,URL: http://www.openmobilealliance.org/

[PoC_XDM-V1_0 TS] "PoC XDM Specification", Version 1.0, Open Mobile Alliance™, OMA-TS-PoC_XDM-V1_0, URL: http://www.openmobilealliance.org/

[Presence_SIMPLE-V1_0 TS] "Presence XDM Specification", Version 1.0, Open Mobile Alliance™, OMA-TS-Presence_SIMPLE-V1_0, URL: http://www.openmobilealliance.org/

[RFC0959] "FILE TRANSFER PROTOCOL (FTP)", J. Postel and J. Reynolds, October 1985, URL: http://www.ietf.org/rfc/rfc0959.txt

[RFC 2753] "A Framework for Policy-based Admission Control", R. Yavatkar et al, January 2000, URL: http://www.ietf.org/rfc/rfc2753.txt

[RFC 3060] "Policy Core Information Model -- Version 1 Specification", B. Moore et al, February 2001, URL: http://www.ietf.org/rfc/rfc3060.txt

[RFC 3198]	"Terminology for Policy-Based Management", A. Westerinen et al, November 2001, URL: http://www.ietf.org/rfc/rfc3198.txt
[RFC 3460]	"Policy Core Information Model (PCIM) Extensions", B. Moore, Ed., January 2003, URL: http://www.ietf.org/rfc/rfc3460.txt
[SAML2.0]	"Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", Scott Cantor et al, OASIS Standard, 15 March 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
[SAML2.0profileXACML2.0]	"SAML 2.0 profile of XACML v2.0", Anne Anderson et al, February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf
[simple-presence-rules]	"Presence Authorization Rules", J. Rosenberg, June 2005, URL: http://www.jdrosen.net/papers/draft-ietf-simple-presence-rules-03.txt
[SIMPLE XCAP]	"The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", J. Rosenberg, June 2005, URL: http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-07.txt
[XACML]	"XACML - eXtensible Access Control Markup Language", OASIS, URL: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20
[XDM Core 1.0 TS]	"XML Document Management (XDM) Specification", Version 1.0, Open Mobile Alliance™, OMA-TS-XDM_Core-V1_0, URL: http://www.openmobilealliance.org/
[XDM Shared 1.0 TS]	"Shared XDM Specification", Version 1.0, Open Mobile Alliance™, OMA-TS-XDM_Shared-V1_0, URL: http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purposes of the present document, the terms and definitions given in [OMA-DICT] and the following apply:

Delegate	A delegate is a designated resource that performs specified tasks or functions on behalf of (one or more) other resources. To *delegate* is to designate a resource to perform specified tasks or functions on behalf of (one or more) other resources.
Policy	An ordered combination of policy rules that defines how to administer, manage, and control access to resources [Derived from [RFC 3060], [RFC 3198] and [RFC 3460]].
Policy Action	Action (e.g. invocation of a function, script, code, workflow) that is associated to a policy condition in a policy rule and that is executed when its associated policy condition results in "true" from the policy evaluation step.
Policy Condition	A condition is any expression that yields a Boolean value.
Policy Enforcement	The process of executing actions, which may be performed as a consequence of the output of the policy evaluation process or during the policy evaluation process.
Policy Evaluation	The process of evaluating the policy conditions and executing the associated policy actions up to the point that the end of the policy is reached.
Policy Management	The act of describing, creating, updating, deleting, provisioning and viewing policies.
Policy Processing	Policy evaluation or policy evaluation and enforcement
Policy Rule	A combination of a condition and actions to be performed if the condition is true
Request	An articulation of the need to access a resource (e.g. asynchronous events).
Requestor	Any entity that issues a request to a resource.
Resource	Any component, enabler, function or application that can receive and process requests.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in [OMA-DICT] and the following apply:

LDAP	Lightweight Directory Access Protocol
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PEF	PEEM/Policy Evaluation and Enforcement
PM	PEEM/Policy Management
PoC	Push to talk over Cellular
PV	PEEM/Policy Evaluation
PF	PEEM/Policy Enforcement
XCAP	XML Configuration Access Protocol

XDM	XML Document Management
XPATH	XML Path Language

4. Introduction (Informative)

Service environments where different entities, e.g. enterprises, network operators, service providers and 3rd party service providers collaborate to provide highly personalised services to subscribers present new opportunities and benefits to the industry. The specification of the Policy Evaluation, Enforcement and Management (PEEM) enabler is driven by the need to reduce management complexity whilst introducing consistent new subscriber services with the same or reduced time to market.

Policies are formalisms that are used to express business, engineering or process criteria represented by a combination of policy conditions and actions. PEEM specifies ways to convey and enforce policies that can be used to manage resources, processes, and underlying systems. OMA enablers are expected to re-use the PEEM specifications in order to avoid duplication and misalignment. The aim of this document is to define the architecture of the PEEM enabler.

PEEM also enables the delegation of responsibility to other resources:

- This can help avoid the costly duplication of functionality across service enablers and reduce the proliferation of 'silos' in service provider domains.
- This is expected to be an efficient mechanism to re-use resources by providing a systematic way to express and implement the delegation to such other resources.

Policies can be associated with target resources, and/or requestors and/or requests/responses; this list is not exhaustive.

OSE dictates that whenever requests are made to a resource, the associated policies are evaluated and enforced by a policy enforcement mechanism on the request and/or on the associated response. The PEEM enabler can be used to perform these operations.

The PEEM enabler can be used as a function that can be explicitly called by other resources or act as a proxy..

This architecture document is neutral in terms of implementation and deployments.

4.1 Planned Phases

All the PEEM requirements are planned to be fully met in this release. No future releases are currently planned.

4.2 Security Considerations

The PEEM enabler can be applied according to two usage patterns (callable and proxy usage pattern). In both usage patterns, interaction with the PEEM enabler implementation may be within the same domain or between different domains, and the resources that interact with the PEEM enabler implementation may differ per usage pattern;

- in the proxy usage pattern the Target Resource Requestor may reside in the same domain as the PEEM enabler domain and security measures should be considered that allow for secure intradomain exchanges. Alternatively the Target Resource Requestor may reside in a different domain from the PEEM enabler domain hence security measures should be considered that allow for secure interdomain exchanges. The intercepted Target Resource Requests may need security, e.g. web services security.
- in the callable usage pattern the PEEM enabler can be explicitly called by an Evaluation Requestor (see Section 5) that may reside in the same domain as the PEEM enabler domain and security measures should be considered that allow for secure intradomain exchanges between Evaluation Requestor and PEEM enabler. Alternatively the Evaluation Requestor may reside in a different domain from the PEEM service enabler domain hence security measures should be considered that allow for secure interdomain exchanges between Evaluation Requestor and the PEEM enabler.

Note that different domains may imply: different administrative domains, different security domains and/or the need to traverse insecure networks between the domains.

In both usage patterns the PEEM service enabler may delegate to (i.e. make a request to) other enablers such as a charging enabler. These other (delegated to) enablers may or may not reside in different security or

administrative domains and appropriate security measures should be considered for each case. In particular, it is important to be able to ensure that the different systems (PEEM, delegated resources and target resources) get access only to the information that they need to know to perform their functions (e.g. payment details are not made available to authentication resource etc.). Appropriate key management and selective encryption when delegating functions may be required and may be specified by the policies.

In both usage patterns the PEEM enabler policies are managed (i.e. create, modify, view, delete policies) through the management interface PEM-2. Various management actors such as network operator and end-user (i.e. Management Requestors, see Section 5) must be supported and appropriate associated security measures need to be applied: it should be possible to authenticate Management Requestors (see section 5), e.g. principles authorised by service provider or third party or an end user) and secure the PEM-2 exchanges for both the intradomain and the interdomain case.

5. Architectural Model

PEEM Architecture is defined by its interfaces and its behavior is driven by the policies written in a policy expression language. PEEM specifications include two interfaces (one for management and one for policy processing) and a policy expression language. With respect to the interactions between PEEM and other resources, two PEEM usage patterns have been identified (callable usage pattern and proxy usage pattern). PEEM architecture diagram, PEEM behavior in different usage patterns and PEEM components, interfaces and policy expression language will be described in the following sections.

5.1 Dependencies

No dependencies identified.

5.2 Architectural Diagram

This section contains architectural diagrams that illustrate PEEM logical components, interfaces and relationships it has with other entities in the OSE.

This document will describe in further detail the logical components and interfaces from the PEEM enabler perspective.

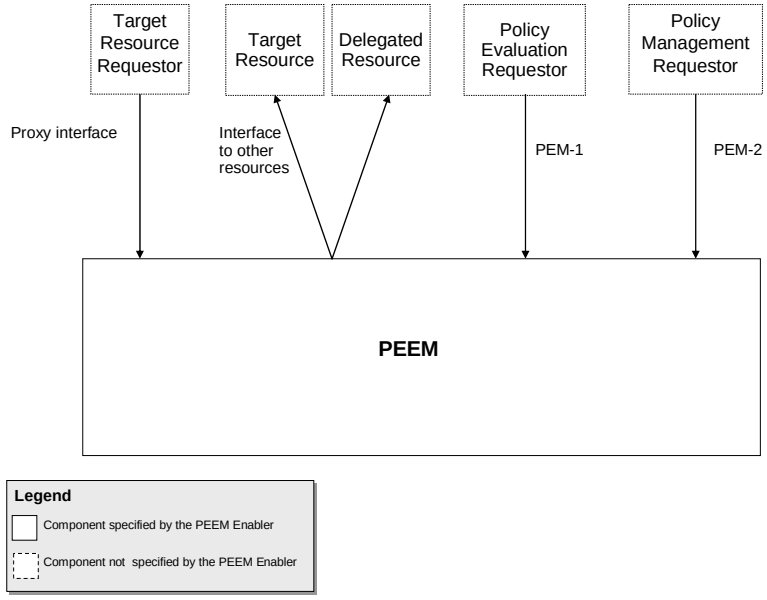

Figure 1. PEEM Enabler architecture

Note that PEEM enabler implementation can evaluate and enforce policies when exposing any resource (e.g. application, enabler, component, function).

For both proxy and callable usage patterns, PEEM may interact with other resources.

A proxy interface supports the PEEM proxy usage pattern (see Figure 2).

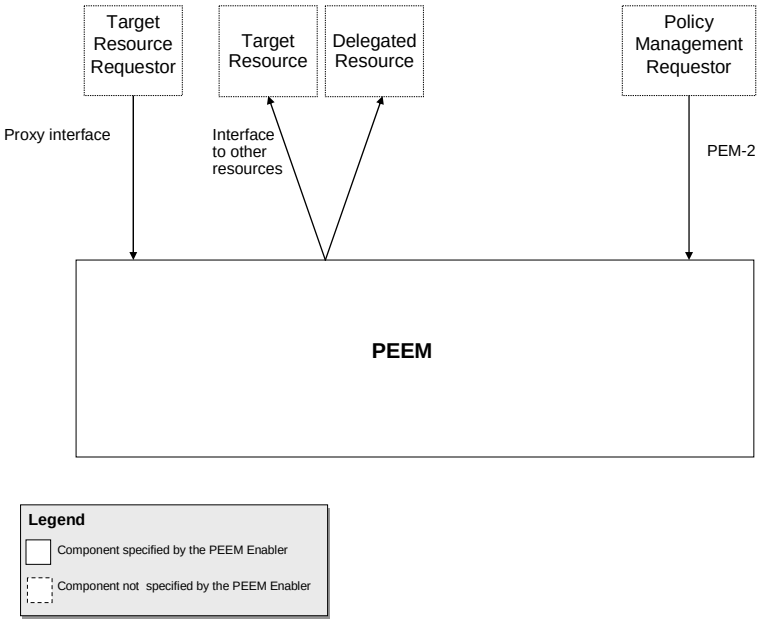

Figure 2. PEEM enabler proxy usage pattern

In the callable usage pattern (see Figure 3), PEEM can act as a Policy Decision Point (PDP), as described in the IETF PEP-PDP model [RFC 2753] (see also section 5.7 and Appendix G).

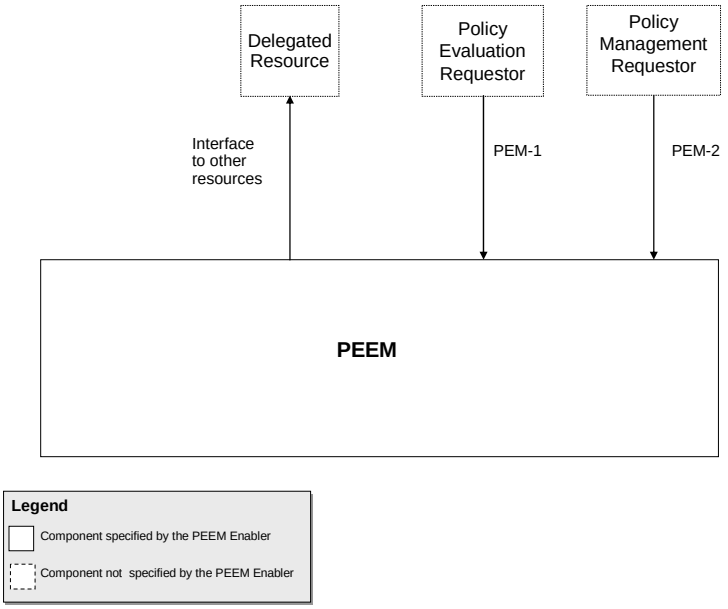

Figure 3. PEEM enabler callable usage pattern

5.3 Functional Components and Interfaces

This section describes the functional components and interfaces identified in Figure 1. The components and interfaces specified by PEEM are loosely coupled, in the sense that the specification for each of them does not have to be tightly coupled with the specification of the others.

The PEEM enabler exposes the following interfaces:

- PEM-1 (PEEM specified callable interface)
- PEM-2 (PEEM specified management interface)
- Proxy interface (used for intercepting requests to target resources)

5.3.1 PEEM (Policy Evaluation, Enforcement and Management component)

PEEM has the following features:

- identifies the policies associated with the request. Policies contain policy rules (see definitions for Policy and Policy Rule)
- processes the policy, i.e. goes through the following steps:
    - evaluates policies using messages received and other context information (see definition for Policy Evaluation, Policy Rule and Policy Condition). The component may delegate to other resources where appropriate
    - executes the policy actions resulting from a positive evaluation of the policy conditions. The component may use delegation to other resources where appropriate, and
    - may return, after completing all previous processing, a policy decision to a requestor or may allow a request to continue to its original target destination. A request for policy processing can arrive to PEEM either as a direct request for support from another entity (see also the section describing PEEM specified callable interface) or as a request from another entity to another resource, proxied (or intercepted) by PEEM (see also the section describing PEEM proxy usage pattern). In the first case, the policy processing may complete by returning a policy decision to the requesting resource (See also 5.7.1 and 5.7.4) or perform enforcement itself (see also 5.7.5). When a policy decision is returned to a resource, that resource is in control of deciding how to handle the rendered decision. In the second case, the policy processing completes by forwarding the original request to the destination resource (if the processing resulted into a “pass” condition) or by returning an error to the originating entity (if the processing resulted into a “fail” condition)
- provides the functions of describing, creating, updating, deleting, provisioning and viewing of policies.

5.3.2 Proxy Interface

The Proxy interface is not specified by PEEM, but is used to exchange messages compliant to the target enablers or more generally messages compliant to combination of the target resource interface and the set of parameters that must be added to requests through that resource’s interface, as required to satisfy policies that are to be processed when exposing the resource. The messages exchanged through this interface may be different for each target resource.

5.3.3 PEM-1 (PEEM specified callable interface)

The PEM-1 interface is used by other resources to make a direct request for policy processing. PEEM processes the request and may return a policy decision (the result of the policy evaluation) to the originating resource, using the same interface, see also 5.7.4. Alternatively, it may also perform policy enforcement and possibly return no value to the requester, see also 5.7.5.

PEM-1 is defined in a way that permits other enablers to particularize the PEM-1 interface in order to satisfy their requirements.

5.3.4 PEM-2 (PEEM specified management interface)

The PEM-2 interface is used by other resources to make a request for policy management. This interface is also referred to as PEEM management interface.

5.4 Other components and interfaces

In addition to PEEM components and interfaces, there are other elements represented in Figure 1 for a better understanding of the architectural diagram. The following is a list of other elements identified in Figure 1 that interact with PEEM:

- Target Resource Requestor
    - Target Resource Requestor represents a resource (e.g. application, enabler) that issues a request to a target resource.
- Target Resource
    - Target Resource represents the destination resource for a request made by another resource.
- Delegated Resource
    - Delegated Resource represents the resource to which PEEM may delegate certain policy actions during the policy processing process.
- Evaluation Requestor
    - Evaluation Requestor represents a resource (e.g. application, enabler) that issues a request for policy processing to PEEM.
- Management Requestor
    - Management Requestor represents a resource (e.g. application, enabler) that issues a request for policy management to PEEM.
- Interface to other resources
    - The interface to other resources is not specified by PEEM, but is used to exchange messages compliant to the interface of the target or delegated enablers or more generally messages compliant to the target or delegated resource interfaces. The messages exchanged through this interface may be different for each resource. The “Proxy Interface” and the “Interface to other resources” have similar properties and behavior.

5.5 Flows

PEEM usage patterns are introduced in Section 5.2. This section describes the high-level logical flows for the PEEM proxy usage pattern and the PEEM callable usage pattern. In addition, this section describes the policy management flow.

5.5.1 PEEM Proxy Usage Pattern Flow

Figure 4 illustrates the logical flows of the PEEM enabler in the proxy usage pattern.

In the PEEM proxy usage pattern the Target Resource Requestor issues a request to the Target Resource (flow #1).

The request is intercepted by the PEEM enabler (acting as a Target Resource proxy in the proxy usage pattern). Upon interception of the request the PEEM enabler identifies the relevant policy and starts the process of evaluating and enforcing it. In that process it may issue requests to one or more Delegated Resources that perform certain expected functions (flow#2) and deal with the results of the delegated functions that are returned to the PEEM enabler (flow#3). Based on a policy that evaluates the returned results the PEEM enabler may again issue requests to one or more Delegated Resources (flow#4) and deal with the results (flow#5).

The Target Resource returns a result (flow#7) which is intercepted by the PEEM enabler (acting as a Target Requestor proxy in the proxy usage pattern). Upon interception of the result (flow#7) the PEEM enabler identifies the relevant policy and starts the process of evaluating and enforcing it. In that process it may issue requests to one or more Delegated Resources that perform certain expected functions (flow#8) and deal with the results that are returned to the PEEM enabler (flow#9). Based on the policy that evaluates the returned results the PEEM enabler may again issue requests to one or more Delegated Resources (flow#10) and deal with the results (flow#11).

The PEEM enabler passes the result, if appropriate, on to the Target Resource Requestor (flow#12).

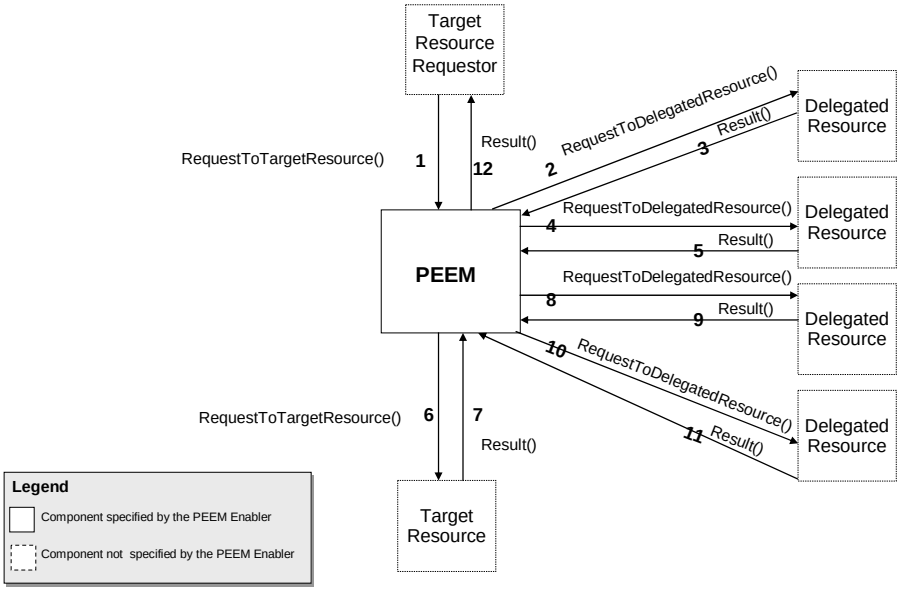

Figure 4. Logical Flow for PEEM proxy usage pattern

5.5.2 PEEM Callable Usage Pattern Flow

Figure 5 illustrates the logical flows of the PEEM enabler in the callable usage pattern.

In the PEEM callable usage pattern the Evaluation Requestor issues a request for Policy processing (flow #1) to the PEEM enabler using the PEM-1 interface.

Upon reception of the request the PEEM enabler identifies the relevant policy and starts the process of evaluating it. In that process it may issue requests to one or more Delegated Resources that perform certain expected functions (flow#2) and may deal with the results (flow#3) that are returned to the PEEM enabler. Such delegated resources can be enablers or other resources.

A decision is reached when the policy evaluation completes. The PEEM enabler then returns the decision (flow #4) to the Evaluation Requestor or performs enforcement itself, possibly without returning a value to the requester, see also 5.7.4 and 5.7.5. Upon reception of the decision the Evaluation requestor executes its own actions as dictated by the decision.

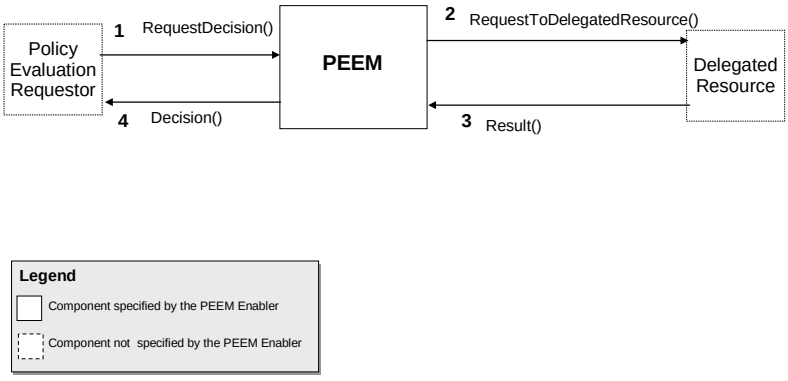

Figure 5. Logical Flow for PEEM callable usage pattern

5.5.3 PEEM Policy Management Flow

Figure 6 illustrates the logical flows of the PEEM enabler for management of policies.

In the PEEM management flow the Management Requestor issues a request for Policy Management (flow #1 in Figure 6) to the PEEM enabler, through the PEM-2 interface. Upon reception of the request the PEEM enabler identifies the type of policy management request (e.g. create, delete, view, modify), executes the appropriate function and returns the results to the Management requestor (flow #2 in Figure 6).

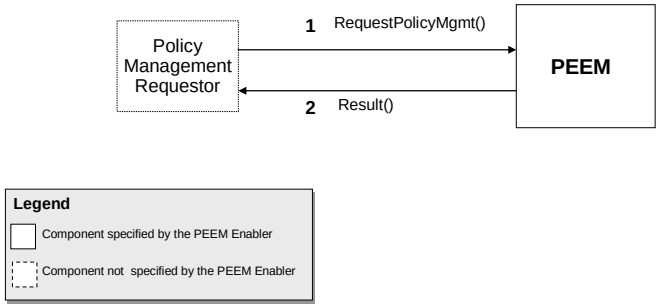

Figure 6. Logical Flow for PEEM policy management

5.6 Policy Expression Language

5.6.1 PEEM Policy information model

By definition, policies are combinations of policy rules, each of which is defined as a policy condition and actions (i.e., IF condition THEN action).

Evaluation may involve arbitrary computations. The conditions and actions in policy rules may require the execution of arbitrary functions that include delegation to OMA enabler implementations.

The topology of a policy is defined as a graph where each node represents a condition to be evaluated and each outbound branch has actions to be executed if the corresponding condition is true. This is illustrated in Figure 7.

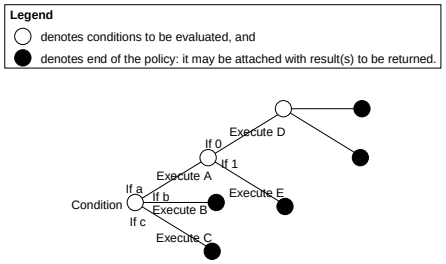

Figure 7. An Example of a graph that describes the topology of a policy (composed of multiple policy rules).

There are 2 execution models described by the IETF policy model defined in RFC 3060 [RFC 3060], one model has a single condition at each node. The second model permits case statements on the nodes where each includes a priority that determines the order of evaluation of these simple conditions.

As mandated in [PEEM-RD], PEEM can also be used to perform only policy evaluation without executing any policy actions or performing any enforcement. The case of a simple condition is illustrated by Figure 8

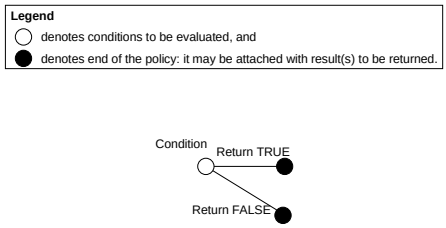

Figure 8. Example of evaluation only policy without execution of any policy actions or enforcement by PEEM

Note again that branching from a node is not limited to 2 branches (e.g. “case of”).

The topology of a policy graph can be changed in numerous ways without changing the result of its evaluation or enforcement. This may of course modify the policy conditions and actions from one graph to another equivalent graph.

The policy to be used by PEEM for any invocation of PEM-1 may be provided as part of the call to PEEM by passing the policy (or a reference) that must be processed.

5.6.2 Properties of an appropriate policy expression language

A suitable policy expression language must satisfy the following requirements:

- It is capable of expressing any combination of policy conditions and actions. In particular:
    - It is powerful enough to specify any calculation within a condition or an action
    - It can support delegation
    - Can perform pattern matching on input data
    - Can specify the format of output data
- Does not preclude any policy topology.

To clarify further, the language may be more easily adopted and deployed if the following options are provided:

- It provides constructs (e.g. function call) to facilitate interface transformation or generation of a new binding.
- It is capable of expressing OMA existing and/or future policy conditions and actions such as (for example):
    - Encryption strength must be 128kbits
    - Authentication is required

- Rating must be checked and charging performed before passing the request
- Subscribers must have a minimum amount in their account

5.7 Mapping IETF PEP-PDP model to the PEEM model (informative)

5.7.1 Essence of the PEP/PDP behavior

Considering the information contained in the sections above, we can conclude that the core of the PEP/PDP behavior that should be supported by PEEM, is the following:

PEP behavior:

- Identifying requests that need an external authorization decision
- Ability to request for external authorization decision
- Enforcing the decision taken in the external authorization function

PDP behavior:

- Receiving a request for taking a decision over an authorization
- Identify relevant policy and take a decision
- Return the decision
- May call delegated resources as part of the evaluation (Not included in the figures below, see section 5.5.2)

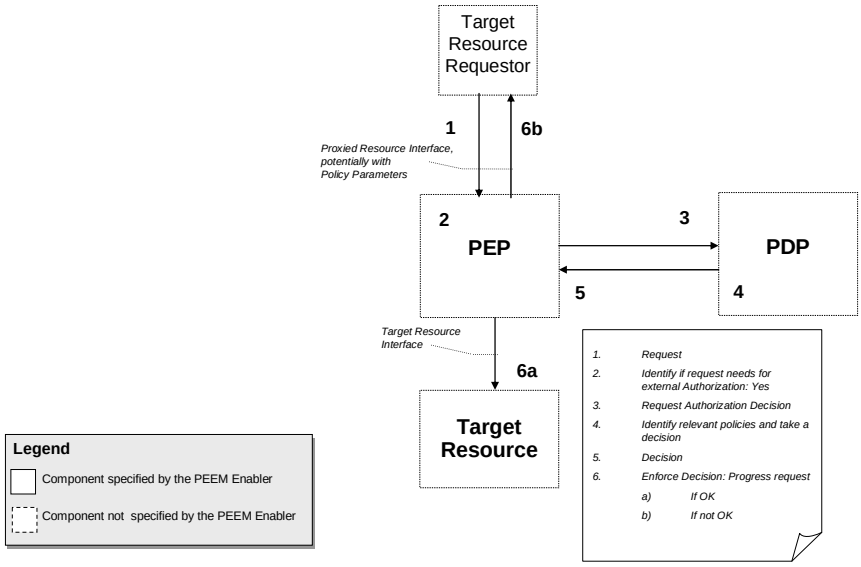

Figure 9. PEP/PDP model behavior

5.7.2 Recap of PEEM model

Utilizing the PEEM model for satisfying the PEP/PDP behavior as shown in previous sections may happen in at least two possible scenarios.

5.7.2.1 Using PEEM for PEP/PDP behavior – Scenario A

In this scenario, both ends of the flow are played by PEEM compliant elements.

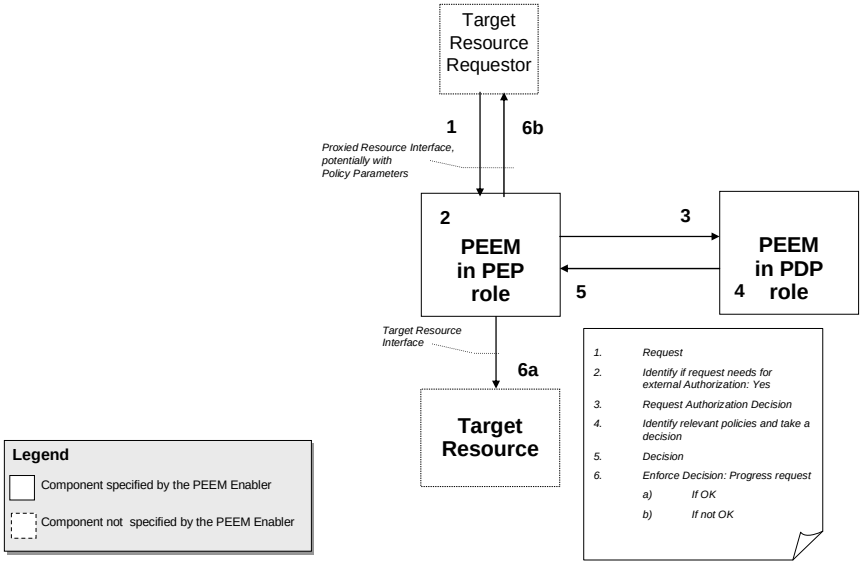

Figure 10. PEEM support for PEP/PDP behavior - Scenario A

When the proxy usage pattern is used, the complete policy enforcement cycle may be realized within one particular PEEM implementation. The case where one particular PEEM implementation provides the PEP and PDP roles at the same time is shown in Figure 11.

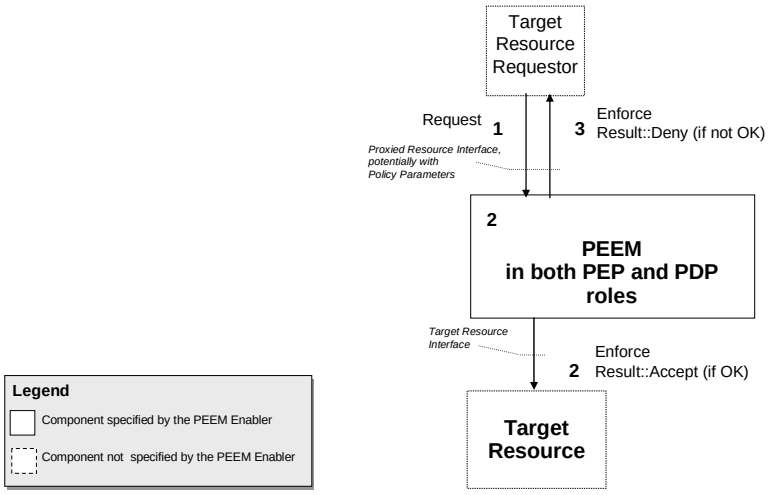

Figure 11 - PEEM support for PEP/PDP behavior in one particular PEEM implementation – Scenario A

Of course, following the guidelines already written in the PEEM AD and PEEM RD, other deployment options may exist for the PEP side, as illustrated in next picture (e.g.: some OMA WGs could decide to fully reutilize the PEEM specifications in their defined enablers); the essence is that the PEP role is played by PEEM functions.

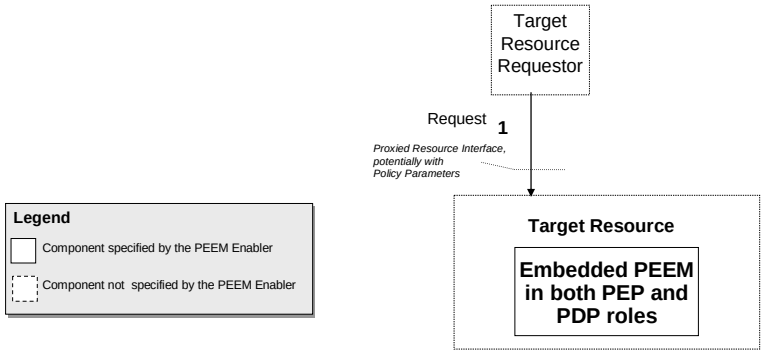

Figure 12. Different deployment options exist on the PEP side

In this case, the functionality required for the PEP and PDP behaviors are supported by PEEM specifications.

5.7.2.2 Using PEEM for PEP/PDP behavior – Scenario B

In this scenario, the PEP functionality is realized by any resource. The mechanisms utilized by this resource to identify and to determine if external authorization is needed are outside the scope of PEEM.

PDP behavior is realized by PEEM.

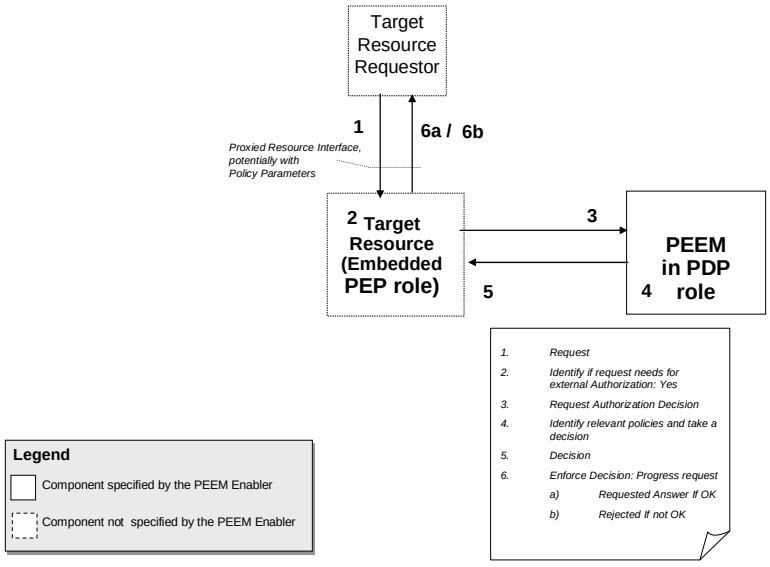

Figure 13. PEEM support for PEP/PDP behavior - Scenario B

The support for the PEP/PDP behavior in this scenario takes place in the following way.

As we said at the beginning of this section, PEP behavior (as stated in Section 5.7.1), is realized by the enabler in a way that is outside of the PEEM spec:

- It could be specified in an OMA enabler spec that reuses PEEM
- It could be done proprietarily by the legacy enabler
- It could be done by an ad-hoc programming
- Etc.

Additionally, please note that in the figure appears the label “Enabler With embedded PEP behavior”, but this could be in fact any kind of requester, e.g.: an application.

The PDP behavior (as stated in Section 5.7.1), is realized by the PEEM enabler based on its specs (interfaces defined, policy expression language defined, etc.)

The communications between both elements take place over the PEM-1 interface, where the decision can be of the nature "accept" or "deny" and in addition it can be of the nature of communicating a more complex decision with additional steps to be undertaken by the PEP (e.g. an outgoing message like "ask user consent").

5.7.3 Support for PEP/PDP behaviors: Impacts on PEEM model

Scenario A

In this case, the situation comes down to one PEEM implementation (realizing the PEP role) delegating the decision onto another PEEM implementation (implementing the PDP role):

- PEM-1 interface: Interface and protocol will give support to this “decision delegation” flow.
- PEEM enabler: needs to be able to satisfy PDP and PEP behaviors:
    - From PEP behavior
        - Be able to identify which service requests need an external authorization decision.
        - Be able to delegate the decision to an external PDP element.
        - Being able to enforce the decision that was taken in an external PDP element.
    - From PDP behavior:
        - Be able to receive decision requests when in callable mode.
        - Be able to identify relevant policy on decision requests coming through the PEM-1 interface, “calculate” a decision and return it, without having to enforce the decision (since in this PEP/PDP behavior, the enforcement of the decision would take place on the PEP side).
- PEEM policy expression language:
    - Needs to give support for defining policies for the PDP behavior (oriented to just give a decision).
    - Needs to give support for defining policies for the PEP behavior.
- PEM-2 interface: Interface and protocol will give support to to provision a kind of policies needed to support the PEP-PDP model.

Scenario B

In this case, the requirements/impacts from supporting the PEP/PDP model fall on the following aspects:

- PEM-1 interface: Interface to request a decision using the PEEM enabler.
- PEEM enabler: needs to be able to satisfy PDP behavior:
    - Be able to receive requests in callable mode.
    - Be able to identify relevant policy on requests coming through the PEM-1 interface, “calculate” a decision and return it, without having to enforce the decision (since in this PEP/PDP behavior, the enforcement of the decision takes place on the PEP side).
- PEEM policy expression language:
    - Needs to give support for defining policies for the PDP behavior (oriented to just give a decision).

5.7.4 Mapping IETF PEP-PDP model to PEEM architecture

This section provides an explanation on how the IETF policy architecture maps to the PEEM architecture.

Disclaimer: while portions of the following text are copied from [RFC2753], they have been modified to show the mapping to the PEEM architecture, where appropriate. Taking into consideration the IETF model that is based on PEP and PDP components, the following table and text applies (copied from [RFC2753]):

Table 1. IETF definitions

IETF definitions [RFC3198] [RFC2753]
Policy Decision Point (PDP) A logical entity that makes policy decisions for itself or for other network elements that request such decisions [RFC2753].
Policy Enforcement Point (PEP) A logical entity that enforces policy decisions [RFC2753]. The PEP enforces the policy decision by appropriately accepting or denying the request [RFC2753].

The basic interaction between the components begins with the PEP. The PEP (in Figure 14 the PEP is Evaluation Requestor) will receive a notification or a message that requires a policy decision. Given such an event, the PEP then formulates a request for a policy decision and sends it to the PDP (see Figure 14, PEM-1, PEP is Evaluation requestor for evaluation only). The PDP returns the policy decision (see Figure 14, PEM-1 to Evaluation requestor) and the PEP then enforces the policy decision by appropriately accepting or denying the request (Evaluation Requestor will continue its processing, in case of an accepted request). The PDP itself may make use of additional mechanisms and protocols to achieve additional functionality such as user authentication, accounting, policy information storage, etc. For example, the PDP is likely to use an LDAP-based directory service for storage and retrieval of policy information (see Figure 14, delegation via interfaces to other (Delegated) Resource).

Thus Figure 14 illustrates the case where the PEEM enabler performs evaluation only [PEEM-RD]. In the case of evaluation only, a PDP type of function is the only one involved. Typically this applies to a PEEM enabler in callable mode. A PEP (Evaluation Requestor) may request evaluation from various delegated PDPs (e.g. from a charging PDP, from a privacy PDP, from a regulatory PDP). The requestor (PEP) will then carry out the decision rendered by the PDP. The components that are “whited out” are not involved in the interactions, and the interfaces to those components are not used. The “**bold**” interfaces indicate which interfaces are used in this mapping, relevant to the PEEM architecture.

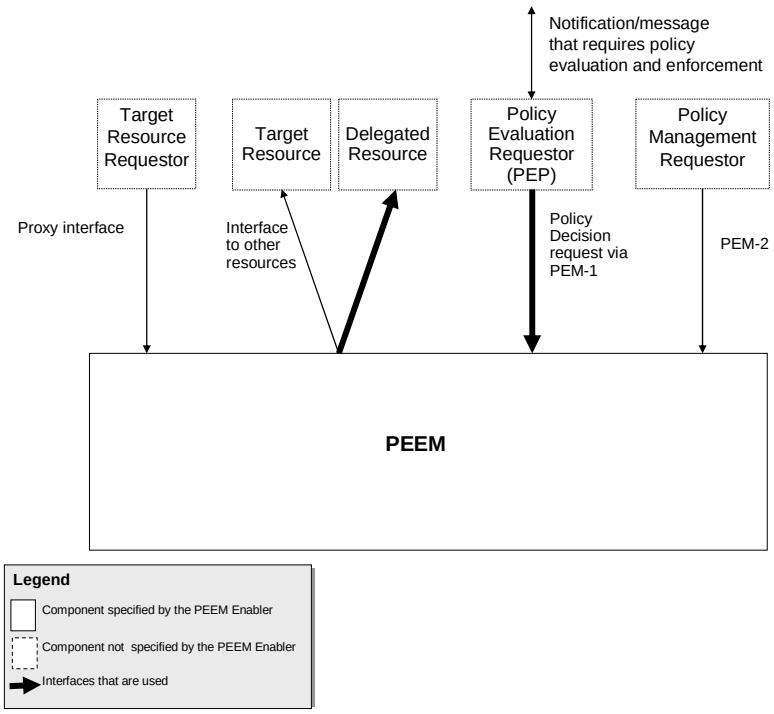

Figure 14. Callable PEEM for evaluation only: PEP-PDP model when only the PDP is incorporated in PEEM

In accordance with the IETF, PEEM may make use of additional mechanisms and protocols to achieve additional functionality. In the case of delegation to OMA enablers such delegations will be done through the enablers I0 interfaces. When the PEP function is performed outside PEEM, that function can also make use of delegation to other resources in the same way as described above.

5.7.5 PEEM features beyond the PEP/-PDP model

PEEM supports usage models beyond what is considered with the PEP-PDP model discussed above:

- The PEEM proxy mode is not modeled by the PEP-PDP model. However, it should be noted that a PEEM enabler implementation for usage in proxy mode may be implemented using PEP interceptors and PDP for policy evaluation.
- In PEEM callable mode, PEEM may support situations where the policies that are evaluated result into:
    - Returning no result to the requester, i.e. the calling resource will delegate the enforcement to PEEM. PEEM then handles the enforcement of the operation via the executed policy actions
    - Returning a result to another resource than the requester, i.e. the calling resource will delegate the enforcement to PEEM. PEEM then calls the other resource as a policy action.

In all these cases, enforcement may be considered to be performed as part of the policy evaluation, delegated by the calling entity/PEP. This explains why the definitions of policy enforcement in section 3.2 includes that enforcement may be performed by PEEM.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-20120724-A	24 Jul 2012	Status changed to Approved by TP Ref TP Doc# OMA-TP-2012-0278-INP_PEEM_V1_0_for_Final_Approval

Appendix B. Related technologies for policy expression languages (Informative)

The following languages may be able to satisfy most of the conditions identified in Section 5.6 (alphabetically sorted):

- 3GPP TS-29.198-13 – "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Open Service Access (OSA); Application Programming Interface (API);Part 13: Policy management Service Capability Feature (SCF)" [3GPP TS-29.198-13]
- BPEL – Business Process Expression Language [BPEL]
- BPML - Business Process Modeling Language [BPML]
- Common Policy – a document format for expressing privacy preferences [COMMONPOL]
- ISO/IEC 9899, "Programming languages – C" [ISO/IEC 9899], Java [Java]
- XACML - eXtensible Access Control Markup Language [XACML]

Appendix C. Policy Expression Languages and other relevant specifications to be considered when selecting a PEEM policy expression language (Informative)

Various standards bodies have specified resource policies (e.g. enabler policies). The following specifications are examples of specifications that contain ways of expressing policies to describe resource specific policy data:

PoC: OMA TS-PoC_XDM-V1_0 specifies the PoC user access policy expression language [PoC_XDM-V1_0 TS]

Location: 3GPP TS 22.071 specifies a location policy expression language [3GPP TS 22.071]

Charging: 3GPP TS 23.125 specifies a charging policy expression language [3GPP TS 23.125]

Presence: The IETF SIMPLE working group has drafted a presence authorization policy expression language [simple-presence-rules]

Privacy: The IETF GEOPRIV working group has drafted a privacy preference policy expression language [COMMONPOL]

These examples of policy expression languages need to be considered when selecting a policy expression language for the PEEM specification.

The PEEM specification should take into account that a service provider may have defined and deployed policies (e.g. according the examples mentioned above), when defining a policy expression language. Any such PEEM policy expression language needs to support the functions (i.e. semantics) of the existing domain-specific policy languages which would facilitate the use of PEEM while minimizing the efforts needed to support the existing policies.When supporting such policies, there is no requirement that mandates the reformatting of existing policies, neither is there a requirement that prevents reformatting.

Appendix D. Source material for consideration for PEM-2 interface specification (Informative)

Based on the PEEM RD relevant requirements, we need to assess whether these candidates:

- meets the PEM-2 requirements. If not all requirements are met, additional specification development may be needed
- is needed in its entirety, or a subset would satisfy the PEM-2 requirements

D.1 PEM-2 candidate XDM/XCAP

XDM includes 2 specifications, XDM Core [XDM Core 1.0 TS] and XDM Shared [XDM Shared 1.0 TS].

XCAP stands for XML Configuration Access Protocol) [SIMPLE XCAP].

Based on a summary analysis, XDM (or XCAP) seems to meet at least some of the requirements for the PEM-2 interface. The main issue may be how to use or how to extend XDM (or XCAP) in order to meet other requirements.

XCAP supports:

- XML payload (supports any schema)
- HTTP transport
- Uses XPATH-style URLs to target content using PUT, GET and DELETE

D.2 PEM-2 candidate FTP

The use of FTP as protocol [RFC0959][FTPrfcs] exposed by PEM-2 can be considered to support uploading, downloading, deleting and renaming policies. Edit of policies is achieved by downloading, editing and uploading policies.

D.3 PEM-2 candidate SIP Event Notification

There is also a SIP Event Notification (“sip-profile” event package allows SUBSCRIBE-NOTIFY model on XDM documents). This may be useful to meet some of the requirements, but this and any other aspects should be analyzed in detail during the specification development cycle.

Appendix E. Informative details (Informative)

E.1 PEEM decomposition choices

There are multiple alternative choices for PEEM further decomposition possible for PEEM implementations. All figures and text in this appendix are informative-only.

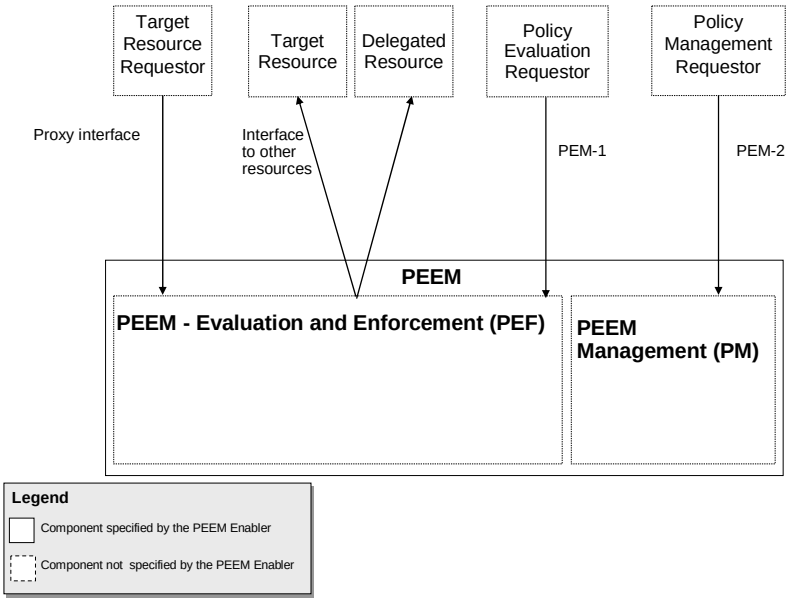

Figure 15. PEEM Enabler architecture

Figure 15 illustrates two logical components, the PEEM Evaluation and Enforcement (PEF) and the PEEM Management (PM) and all interfaces exposed/supported by PEEM.

In Figure 16, PEF is illustrated as decomposed in two distinguishable logical components, PEEM Evaluation (PV) and PEEM Enforcement (PF).

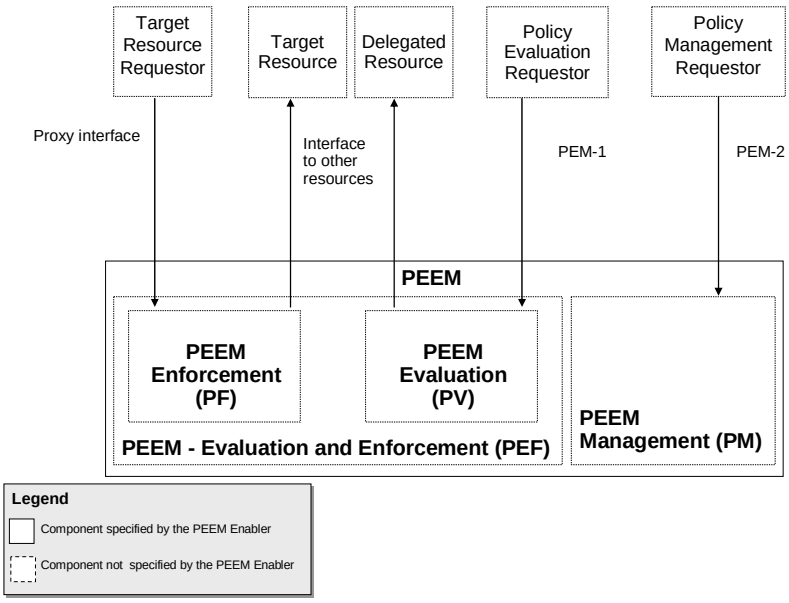

Figure 16. PEEM Evaluation and Enforcement (PEF) illustrated as two logical components, PEEM Evaluation (PV) and PEEM Enforcement (PF)

Thus, the PEEM enabler can be used in callable mode for PV only, to accommodate the requirements for policy evaluation only.

Figure 17 illustrates the case where the PEEM enabler performs evaluation only [PEEM-RD]. Typically this applies to a PEEM enabler in callable mode.

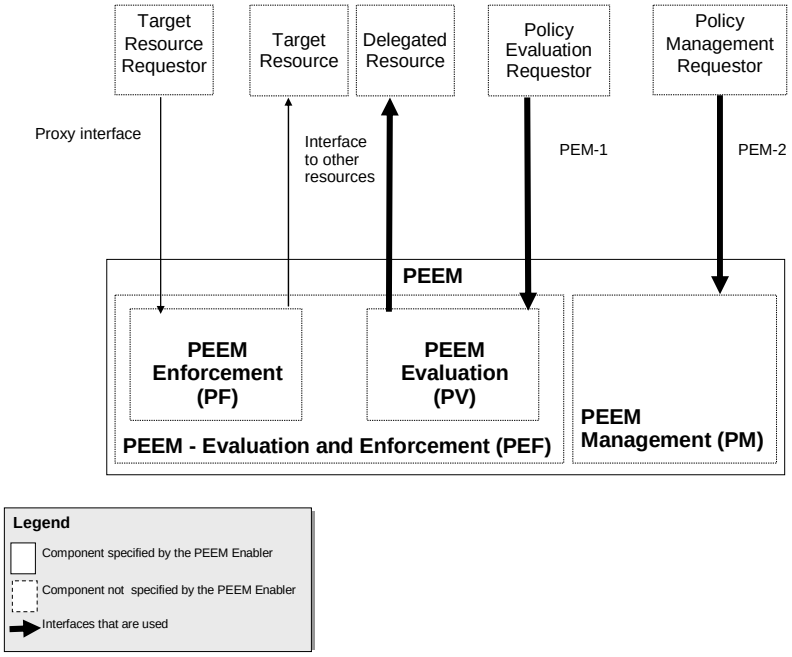

Figure 17. PEEM Enabler – relevant interfaces and components, when the PEEM Enabler performs Evaluation only

The components and interfaces not used in this specific case are shadowed.

NOTE: the PEM-1 interface is used only for evaluation requests, in this case.

E.2 Functional components and interfaces

This section describes the functional components and interfaces identified in Figure 15, Figure 16, and Figure 17. The components and interfaces specified by PEEM are loosely coupled, in the sense that the specification for each of them does not have to be tightly coupled with the specification of the others (e.g. PEF and PM are completely decoupled, PEM-1 and PEM-2 are completely decoupled, PEF and PEM-1 are only coupled by the content of the incoming request, PEM-2 and PM are only coupled by the content of the incoming request, PEF and PEM-2 are completely decoupled and PM and PEM-1 are completely decoupled).

The following is a list of PEEM components (identified because they interact with an interface specified by PEEM):

- PEF (Policy Evaluation and Enforcement component). In the particular case depicted by Figure 15 and Figure 16 PEF is illustrated decomposed in two components:
  - PV (PEEM Evaluation). In the case of evaluation only (see Figure 17), PV is the only component involved.
  - PF (PEEM Enforcement).
- PM (Policy Management component)
- Other entities:
  - PEF Requestor
    - PEF Requestor represents a ressource that issues a direct request for policy evaluation only (PV only) or evaluation and enforcement to the PEEM PEF component [PEEM-RD].
  - PM Requestor

- PM Requestor represents resource that issues a direct request for policy management to the PEEM PEF component [PEEM-RD].

- Other Resources
  - Other Resources represents other resources external to OMA – the subset of such resources as described in the OSE architecture document with which PEEM may need to communicate [OSE-TS].

E.2.1 PEF (Policy Evaluation and Enforcement component)

The PEF (Policy Evaluation and Enforcement) component has the following features:

- identifies the policies associated with the request. Policies contain policy rules (see definitions for Policy and Policy Rule)
- processes the policy, i.e. goes through the following steps:
  - evaluates policies using messages received and other context information (see definition for Policy Evaluation, Policy Rule and Policy Condition). The component may delegate to other resources where appropriate
  - execute the policy actions resulting from a positive evaluation of the policy conditions. The component may use delegation to other resources where appropriate, and
  - may return, after completing all previous processing, a policy decision to a requestor or may allow a request to continue to its original target destination. A request for policy processing can arrive to PEEM either as a direct request for support from another entity (see also the section describing PEEM specified callable interface) or as a request from another entity to another resource, proxied (or intercepted) by PEEM (see also the section describing PEEM proxy usage pattern). In the first case, the policy processing may complete by returning a policy decision to the requesting resource (See also 5.7.1 and 5.7.4) or perform enforcement itself (see also 5.7.5). When a policy decision is returned to a resource, that resource is in control of deciding how to handle the rendered decision. In the second case, the policy processing completes by forwarding the original request to the destination resource (if the processing resulted into a “pass” condition) or by returning an error to the originating entity (if the processing resulted into a “fail” condition)

..

E.2.2 PEF (Policy Evaluation and Enforcement component) with decomposition

The following paragraphs describe the particular case when PEF is decomposed in PEEM Evaluation (PV) and PEEM Enforcement (PF).

PV (PEEM Evaluation component)

The PV (PEEM Evaluation) component is responsible for the policy evaluation portion of the PEEM requirements. This component has the following features:

- identifies the policies associated with the request.,
- evaluates these policies using context information provided by the PEF requestor
- The PV component may use delegation to other resources where appropriate.
- returns, after completing all previous processing, the result of the evaluation to the PEF requestor.

PF (PEEM Enforcement component)

The PF (PEEM Enforcement) component is responsible for the policy enforcement portion of the PEEM requirements. This component has the following features:

- PF performs the "action" as a consequence of the result that was returned by PV (PEEM Evaluation component),

- The PF component may use delegation to other resources where appropriate.

E.2.3 PM (Policy Management)

The PM (Policy Management) component provides the functions of describing, creating, updating, deleting, provisioning and viewing of policies.

Appendix F. Assessment of OMA enabler's policy language needs (Informative)

F.1 PoC User Access Policies

The PoC XDM specification [PoC_XDM-V1_0 TS] specifies PoC User Access Policies.

Policies are XML Documents which are stored on a XDM server. Policies are retrieved from the XDM server using XCAP (e.g. for policy management purposes).

F.1.1 Properties of PoC User Access Policies

The PoC User Access Policy document SHALL conform to the structure of the policy document described in [COMMONPOL], which means that a <ruleset> (synonym for policy) can contain zero or more <rules>. The ordering of the rules is irrelevant.

A <rule> element makes use of the following two elements:

- <conditions>
- <actions>

NOTE: Note that [COMMONPOL] also defines <transformations> but these can (1) be regarded a type of <action> and (2) are not used in PoC User Access policies.

The policy scheme is extensible towards specific application domains: each domain can add application specific policy conditions and actions.

The conditions part is a set of expressions which evaluates to either TRUE or FALSE

Each rule is equipped with a parameter that identifies the rule.

F.1.2 PoC Condition elements

The <conditions> element supports the following PoC application specific XML elements

- the <identity> element.
- the <external-list> element.
- the <other-identity> element.

F.1.3 PoC Action elements

The <actions> element supports the PoC application specific <allow-invite> element, which defines the action the PoC Server is to take when processing a PoC session invitation for a particular user.

F.2 PAG Authorization Policies

The presence XDM specification [Presence_SIMPLE-V1_0 TS] specifies Presence Authorization Policies.

Policies are XML Documents which are stored on a XDM server. Policies are retrieved from the XDM server using XCAP (e.g. for policy management purposes).

F.2.1 Properties of Presence Authorization policies

All policies follow [simple-presence-rules], which means that an authorization policy is in three parts:

- Conditions ('When to apply the rule')
- Actions ('What to do when the condition applies')
- Transformation ('How the data needs to be modified if this is true'), where the transformation element could be another type of "Action".

F.2.2 Types of Presence Authorization policies

There are two types of authorization policies specified:

- Subscription authorization policies which of which are specified the “action” and “condition” parts of the Subscription Authorization policy
- Presence content policies of which are specified the “transformation” part of the policy.

F.3 Common Denominator of the OMA enabler’s policy needs

The common denominators between PoC and PAG policies are:

- use XML to describe the policy
- distinguish a <rule> that contains a <condition> and an <action>
- use XDM and XCAP to perform management of the policies.
- The policy scheme that the policies are based on allows for extensibility for defining application specific <condition> elements and <action> elements.

Appendix G. IETF PEP-PDP model (Informative)

The informative section contains portions that were copied from the following IETF RFC.

Copyright Notices:

For [RFC2753]: Copyright (C) The Internet Society (2000). All Rights Reserved.

> This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The following sections provide an introduction to the IETF policy architecture described in [RFC2753], titled “A framework for policy-based admission control”.

G.1 Introduction to IETF terminology

This section introduces the IETF terminology described in [RFC3198], titled “Terminology for policy-based management” and how the PEEM terminology is related to that.

Table 2. IETF to PEEM terminology mapping

IETF definitions [RFC3198]	Corresponding OMA definitions [PEEM AD]
Policy "Policy" can be defined from two perspectives: A definite goal, course or method of action to guide and determine present and future decisions. "Policies" are implemented or executed within a particular context (such as policies defined within a business unit). Policies as a set of rules to administer, manage, and control access to network resources [RFC3060].	Policy An ordered combination of policy rules that defines how to administer, manage, and control access to resources, [Derived from [RFC 3060], [RFC 3198] and [RFC 3460]].
Policy rule A basic building block of a policy-based system. It is the binding of a set of actions to a set of conditions - where the conditions are evaluated to determine whether the actions are performed [RFC3060].	Policy rule A combination of a condition and an action to be performed if the condition is true
Policy condition A representation of the necessary state and/or prerequisites that define whether a policy rule's actions should be performed. This representation need not be completely specified, but may be implicitly provided in an implementation or protocol. When the policy condition(s) associated with a policy rule evaluate to TRUE, then (subject to other considerations such as rule priorities and decision strategies) the rule should be enforced. In [RFC3060], a rule's conditions can be expressed as either an ORed set of ANDed sets of statements (disjunctive normal form), or an ANDed set of ORed	Policy condition A condition is a Boolean predicate that yields true or false. It may be “complex”.

sets of statements (conjunctive normal form). Individual condition statements can also be negated.	
Policy action Definition of what is to be done to enforce a policy rule, when the conditions of the rule are met. Policy actions may result in the execution of one or more operations to affect and/or configure network traffic and network resources. In [RFC3060], a rule's actions may be ordered.	Policy action Action (e.g. invocation of a function, script, code, workflow) that is associated to a policy condition in a policy rule and that is executed when its associated policy condition results in "true" from the policy evaluation step.
Policy decision Two perspectives of "policy decision" exist: • A "process" perspective that deals with the evaluation of a policy rule's conditions • A "result" perspective that deals with the actions for enforcement, when the conditions of a policy rule are TRUE	Policy evaluation The process of evaluating the policy conditions and executing the associated policy actions up to the point that the end of the policy is reached.
Policy enforcement The execution of a policy decision.	Policy enforcement The process of executing actions, which may be performed as a consequence of the output of the policy evaluation process or during the policy evaluation process.

G.2 Introduction to IETF Policy Architecture

The IETF policy architecture is described in [RFC2753], titled “Framework for policy-based admission control”. Note that [RFC2753] elaborates quite extensively on the policy architecture. The following introductory text is copied from [RFC2753] and some crucial parts have been underlined:

> The two main architectural elements for policy control are the PEP (Policy Enforcement Point) and the PDP (Policy Decision Point). Figure 1 shows a simple configuration involving these two elements; PEP is a component at a network node and PDP is a remote entity that may reside at a policy server. The PEP represents the component that always runs on the policy aware node. It is the point at which policy decisions are actually enforced.
>
> Policy decisions are made primarily at the PDP. The PDP itself may make use of additional mechanisms and protocols to achieve additional functionality such as user authentication, accounting, policy information storage, etc. For example, the PDP is likely to use an LDAP-based directory service for storage and retrieval of policy information.

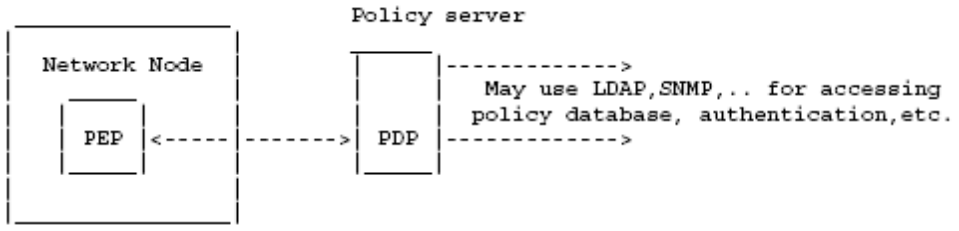

Figure 18. A simple configuration with the primary policy control architecture components. PDP may use additional mechanisms and protocols for the purpose of accounting, authentication, policy storage, etc.

> The basic interaction between the components begins with the PEP. The PEP will receive a notification or a message that requires a policy decision. Given such an event, the PEP then formulates a request for a policy decision and sends it to the PDP. The PDP returns the policy decision and the PEP then enforces

the policy decision by appropriately accepting or denying the request. The PDP may also return additional information to the PEP. This information need not be associated with an admission control decision. Rather, it can be used to formulate an error message or outgoing/forwarded message (e.g. ask user consent).

It must be noted that the PDP acts as the final authority for the decision returned to the PEP and the PEP must enforce the decision rendered by the PDP.

In the case of an external PDP, the need for a communication protocol between the PEP and PDP arises. In order to allow for interoperability between different vendors networking elements and (external) policy servers, this protocol should be standardized.

Appendix H. Source material for consideration for PEM-1 interface specification (Informative)

H.1 PEM-1 candidate SAML profile of XACML

Based on the PEEM RD relevant requirements, a relevant specification is the SAML 2.0 profile of XACML v2.0 specification [SAML2.0profileXACML2.0] – as a potential basis for specifying the PEM-1 interface:

- Chapter 3 of [SAML2.0profileXACML2.0] defines a policy decision query request and policy decision response schema.
- The [SAML2.0profileXACML2.0] policy decision query request allows the Evaluation requestor to submit an XACML request context [XACML, chapter 3.2 and 6] in a SAML request [SAML2.0, Chapter 3] to the callable PEEM enabler.
- The request context in the policy decision query request of [SAML2.0profileXACML2.0] allows for transport of input data for the policy evaluation and/or enforcement process.
- The [SAML2.0profileXACML2.0] policy decision response allows the PEEM enabler to return an XACML decision response context [XACML, chapter 3.2 and 6] in a SAML response [SAML2.0, Chapter 3] to the Evaluation requestor. The Evaluation Requestor enforces that decision.
- The response context in the policy decision response of [SAML2.0profileXACML2.0] allows for transport of a decision as the result of a policy evaluation and/or enforcement process.
- The [SAML2.0profileXACML2.0] supports XML payload.
- The [SAML2.0profileXACML2.0] policy decision query request and decision response scheme is extensible.
- The [SAML2.0profileXACML2.0] is extensible to support input and output BLOB parameters.
- The [SAML2.0profileXACML2.0] is optimized for authorization policy decision requests and policy decision responses and can be generalized for generic policy decision requests and policy decision responses.

H.2 PEM-1 candidate BLOB

PEM-1 can have two input parameters and one output parameter:

- A BLOB input parameter that can carry any binary data. An example of BLOB interface can be found at [J2SEBLOB].
- An optional input parameter specifying the policy to enforce.
- A BLOB output parameter that can carry any binary data, its contents and format defined by the policy rules. To respect its contract with authorized requestors, policies should return the expected data types

Note that the same interface is used for both evaluation or evaluation and execution.

H.3 PEM-1 candidate Parlay Policy Management

[3GPP-OSA-policy-mgmt] is a 3GPP joint Parlay/OSA specification that includes definition of policy expression language aspects, as well as definition for interfaces and API. This specification is therefore considered a valid candidate to be taken into consideration when proposing a technical specification for the PEM-1 (PEEM callable) interface, as it provides an API for Policy Evaluation and a technology to match input context to a policy.