Policy Evaluation, Enforcement and Management Callable Interface (PEM-1) Technical Specification

Approved Version 1.0 – 24 Jul 2012

Open Mobile Alliance

OMA-TS-PEEM_PEM1-V1_0-20120724-A

Contents

Figures

Tables

1. Scope

This document provides the PEEM enabler specification that defines the PEEM interface for requesting PEEM policy processing (PEM-1). The specification supports any kind of input/output needed by the policy, and in addition it describes generic data formats of the interface, PEEM specific input/output parameters , detailed message flows, and mapping of generic data format and message flows to selected protocols. While this specification fully defines the PEM-1 interface, this interface is extensible in the sense that it may be enhanced to support additional input/output parameters defined in other enablers, which plan to re-use the PEM-1 interface specification for their specific purposes. The PEM-1 interface may also be extended outside the OMA, through vendors' and/or Service Providers' customization.

Because such extensibility may result in both mandatory and optional parameters, a Service Provider may be expected to store, publish and/or advertise the details of the supported PEM-1 interface, so that entities that request policy processing know what is expected of them; however, mechanisms for storing, publishing and/or advertising the supported PEM-1 specification options are out-of-scope for the PEM-1 specification. Finally, it is also out-of-scope for the PEM-1 specification to define how PEM-1 input parameters are processed by a PEEM enabler implementation, or how PEM-1 output parameters are processed by a resource that requested PEEM policy processing.

2. References

2.1 Normative References

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.6, Open Mobile Alliance™, OMA-IOP-Process-V1_6, URL:http://www.openmobilealliance.org/

[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

[HTTP 1.1] Hypertext transfer Protocol HTTP/1.1", RFC 2616, June 1999

[OMNA] "Open Mobile Naming Authority", Open Mobile Alliance™, URL: http://www.openmobilealliance.org/Technical/OMNA.aspx

[OWSER] "OMA Web Services Enabler", Open Mobile Alliance™, owser_v1_1, URL: http://www.openmobilealliance.org

[PEEM AD] "Policy Evaluation, Enforcement and Management Architecture", Open Mobile Alliance™, OMA-AD_Policy_Evaluation_Enforcement_Management-V1_0, URL: http://www.openmobilealliance.org

[PEEM FAULT WSDL] PEEM WSDL definition file for the PEM-1 Faults, Open Mobile Alliance™, OMA-SUP-WSDL-PEM_1_faults-V1_0, URL:http://www.openmobilealliance.org

[PEEM INPUT TEMPLATE XSD] PEEM XSD definition file for Policy Input Template, Open Mobile Alliance™, OMA-SUP-XSD_PEM_1_GenericInputTemplateData-V1_0, URL: http://www.openmobilealliance.org/

[PEEM OUTPUT TEMPLATE XSD] PEEM XSD definition file for Policy Output Template, Open Mobile Alliance™, OMA-SUP-XSD_PEM_1_GenericOutputTemplateData-V1_0, URL: http://www.openmobilealliance.org/

[PEEM REQ WSDL] PEEM WSDL definition file for the PEM-1 Request Interface, , Open Mobile Alliance™, OMA-SUP-WSDL-PEM_1_REQ-V1_0, URL:http://www.openmobilealliance.org/

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

[PEEM RSP WSDL] PEEM WSDL definition file for the PEM-1 Response Interface, Open Mobile Alliance™, OMA-SUP-WSDL-PEM_1_RSP-V1_0, URL:http://www.openmobilealliance.org/

[RFC 793] "Transmission Control Protocol", DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION prepared by the Information Sciences Institute, University of Southern California, September 1981, URL:http://www.faqs.org/rfcs/rfc793.html

[RFC 2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC 2396] "Uniform Resource Identifiers (URI): Generic Syntax", Berners-Lee, T., Fielding, R. and L. Masinter, August 1998, URL: http://www.rfc-editor.org/rfc/rfc2396.txt

[RFC 2960] "Stream Control Transmission Protocol", Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, October 2000, URL: http://www.rfc-editor.org/rfc/rfc2960.txt

[RFC 3309] "Stream Control Transmission Protocol (SCTP) Checksum Change", Stone, J., Stewart, R., Otis, D., September 2002, URL: http://www.rfc-editor.org/rfc/rfc3309.txt

[RFC 3588] "Diameter Base Protocol", Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko,

September 2003, URL: http://www.rfc-editor.org/rfc/rfc3588.txt

[RFC 5224] "Diameter Policy Processing Application", Brenner, M., March 2008, URL: http://www.rfc-editor.org/rfc/rfc5224.txt

[SOAP 1.1] "Simple Object Access Protocol (SOAP 1.1)", W3C Note, 08 May 2000, URL: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP 1.2] "SOAP Version 1.2 Part 0: Primer", W3C Recommendation , June 24 2003, URL: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[URI] "Uniform Resource Identifier (URI): Generic Syntax", Fielding, R. and Masinter, L. January 2005, URL: http://www.rfc-editor.org/rfc/rfc3986.txt

[USER NAME TOKEN PROFILE 1.0] Web Services Security Username Token Profile 1.0 OASIS Standard 200401, March 2004, URL: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0

[WSDL 1.1] "Web Services Description Language (WSDL) Version 1.1", W3C Note March 15 2001 , URL: http://www.w3.org/TR/wsdl

[WS Security 1.0] Web Services Security v1.0 (WS-Security 2004) OASIS Standard 200401, March 2004, URL: http://www.oasis-open.org/specs/index.php#wssv1.0

[X.509 TOKEN PROFILE 1.0] Web Services Security, 3 X.509 Certificate Token Profile, 4 OASIS Standard 200401, March 2004, URL: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0

2.2 Informative References

[3GPP TS 29.229] 3GPP TS 29.229, V7.5.0, 2007-03, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Cx and Dx Interfaces based on the Diameter protocol; protocol details (Release 7)"

[3GPP TS 29.329] 3GPP TS 29.329, V7.3.0, 2006-09, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Sh Interface based on Diameter protocol (Release 7)"

[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

[RFC 4234] "Augmented BNF for Syntax Specifications: ABNF", Crocker D., Ed. et al, October 2005, URL: http://www.rfc-editor.org/rfc/rfc4234.txt

3. Terminology and 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 [RFC 2119].

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

3.1 Definitions

Binary Large OBject	An object that can hold any digitized information. A Binary Large Object (BLOB) does not define a specific data structure but instead can convey / contain any data structure.
Custom PEM-1 Template	A PEM-1 Template specified outside OMA (e.g. by Service Providers). A Custom PEM-1 Template is composed from one or more PEM-1 Templates.
Custom PEM-1 Template Binding	A PEM-1 Template Binding specified outside OMA (e.g. by Service Providers).
External Policy	A policy made available to a PEEM implementation as a PEM-1 Parameter that is part of a PEM-1 request.
Internal Policy	A policy made available to a PEEM implementation using the PEEM Policy Management interface (PEM-2), prior to a PEM-1 request to make use of the policy.
PEM-1 BLOB Parameter	A protocol independent PEM-1 interface parameter that is passed as a BLOB between a PEEM requestor and a PEEM enabler implementation.
PEM-1 Input BLOB Parameter	A PEM-1 BLOB Parameter sent by a PEEM requestor to a PEEM enabler implementation, as part of a policy evaluation request message.
PEM-1 Output BLOB Parameter	A PEM-1 BLOB Parameter sent by a PEEM enabler implementation to a PEEM requestor, as part of a policy evaluation response message.
PEM-1 Template Binding	A PEM-1 Template representation associated with a specific technology.
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.
Standard PEM-1 Template	A PEM-1 Template specified by OMA. A Standard PEM-1 Template is composed from one or more PEM-1 Templates.
Standard PEM-1 Template Binding	A PEM-1 Template Binding specified by OMA.

3.2 Abbreviations

ABNF	Augmented Backus–Naur Form
AVP	Attribute Value Pair
BLOB	Binary Large Object
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
PDA	Policy-Data-Answer
PDR	Policy-Data-Request
PEEM	Policy Evaluation, Enforcement and Management
PEL	(PEEM) Policy Expression Language
PEM-1	PEEM Callable interface
PEM-2	PEEM Management interface
OMA	Open Mobile Alliance
RPC	Remote Procedure Call
SCTP	Stream Control Transmission Protocol
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
WSDL	Web Services Definition Language
XML	eXtended Markup Language
XSD	XML Schema Definition

4. Introduction

The PEM-1 (PEEM callable interface) specification is defined to support exchanges between requestors for policy processing and a PEEM enabler implementation. Since there is great variability amongst resources, it is expected that policies will be quite diverse, and therefore may require different input information to be supplied with each request, and in turn, may provide different output information in response. As a consequence, the input and output parameters that may be passed to/from a PEEMenabler implementation may vary greatly with the policy being processed. PEEM requirements [PEEM RD] include the need to define an interface through which resources can invoke policy processing. Having an interface separately defined for each specific policy is impractical, because it implies a difficulty in implementation for both the PEEM enabler implementation, as well as for the resources that may invoke policy processing. The solution to this is described in PEEM architecture [PEEM AD] which defines PEM-1 as a generic interface, that has a well-defined structure of requests and responses, where each request or response message always carries, from the protocol perspective a binary large object, encompassing the collection of input parameters (in the case of a request message) or the collection of output parameters (in the case of a response message).

The variability, depending on policy, of potential input and output parameters is hidden from the interface, by encoding all input or output parameters in the binary large object, according to a well-defined convention. Such convention is based on standard templates, defined in PEEM, and possibly extended by other enabler releases, and/or on custom templates. The PEM-1 specification defines the generic interface, the input and output binary large object that carry the input/output payload, a number of standard templates that can be encapsulated in such binary large objects, and the binding of request/response messages and input/output templates to specific protocol options. Given the deliberate approach of defining PEM-1 as an extensible interface, it is expected that new templates, and input/output parameters that are part of such templates will continue to be specified either in later PEEM releases, or in other enabler releases, or by vendors and Service Providers, therefore extending PEM-1 to accommodate the specific needs of policies deployed in all domains, or in a specific domain.

The PEM-1 specification is loosely coupled to other PEEM specifications and can evolve independent of the others. There is no coupling between PEM-1 specification and PEM-2 (PEEM management interface) specification. PEM-1 specification and Policy Expression Language (PEL) specification share an interest in supporting the same set of input/output variables. In other words, if PEL specification adds support for certain input/output variables because of specific policies instances, then in order for such instances to be processed, PEM-1 may need to be capable of supporting the corresponding input/output parameters. The PEM-1 interface specification is independent of the PEL used by the policies exchanged across the interface. For the request to operate properly, the policies are expected to be defined using a PEL supported by the PEEM implementation.

5. PEM-1 Interface

5.1 Specification of the PEM-1 interface

5.1.1 BLOB behavior

The PEM-1 interface MUST support a BLOB interface [J2SEBLOB] for input and output:

- Any input can be passed via PEM-1 by a requestor
- Any output may be returned via PEM-1 to the requestor

Inputs are parsed and examined and outputs are generated based on the policy processed by PEEM.Interpretation of the BLOB input data structure and generation of output data is always driven by the policy that is processed. If a PEM-1 Template is used within a BLOB, as described in section 5.1.2, the data structure is expected to follow the PEM-1 Template and the policy is expected to be designed to interpret the data structure accordingly.

There needs to be a way for the requestor to know the BLOB's data structure to send as input and expect as output. This may be for example according to PEM-1 Templates as discussed in sections 5.1.2 and after.

In order to use the PEM-1 interface of PEEM, the requestor is aware of the input it needs to provide, and the output behaviour. How the requestor is made aware of these details, (e.g. the PEM-1 Template to follow and expect) is out of scope of PEEM specifications and it is assumed to be communicated in a separate communication channel. Some informative considerations are provided in Appendix F.

BLOBs allow for any bindings to the interface.

5.1.2 PEM-1 Templates

The policies determine how to interpret the incoming PEM-1 Parameters and how to construct outgoing PEM-1 Parameters. PEM-1 Templates define what input PEM-1 Parameters are expected to be provided as input by the requestor for the consumption by the policy and what output PEM-1 Parameters may be generated by the policy for the requestor's consumption.

PEM-1 Templates are defined to permit binding of the PEM-1 interface to as many of the known bindings as possible.

A PEM-1 Template is Standard, as defined in section 5.1.3, (i.e. defined by OMA and included with the PEEM specifications) or Custom (e.g. defined by the Service Provider which deploys PEEM). PEM-1 Template defines the structure of data that represents input for policy processing, or output from policy processing. The actual input is interpreted by the policy, and the generated output is determined by the policy. Policies and PEM-1 Templates should be designed considering the constraints of the requestor and service provider who defines the policies.

PEM-1 parameters and Standard PEM-1 Templates are specified as part of some OMA enablers. PEEM PEM-1 TS captures those PEM-1 Parameters and Standard PEM-1 Templates which can be reused by other OMA enablers.

5.1.3 Standard PEM-1 Templates

To support a Standard PEM-1 Template means that a policy can interpret the incoming data and/or generate outgoing outputs as defined by the PEM-1 Template. Standard PEM-1 Templates are not mutually exclusive. PEM-1 Templates are expressed independently of the binding to a particular technology. Specific bindings are discussed in section 5.3.

5.1.4 Custom PEM-1 Templates

Custom PEM-1 Templates are outside the scope of the PEEM specification, but provide a similar way for service providers to define SP-specific PEM-1 Templates that would be processed by a set of policies used by the service provider.

Custom PEM-1 Templates are PEM-1 Templates defined by the Service Provider in order to support their specific policy needs. Custom PEM-1 Templates are similar to the Standard PEM-1 Templates and similarly used. They may re-use some or all of the input/output PEM-1 Parameters specified in the Standard PEM-1 Templates, and/or may add new input/output PEM-1 parameters.

Custom PEM-1 Templates are not mutually exclusive, neither are they mutually exclusive when considered in combination with Standard PEM-1 Templates.

5.1.5 Encapsulating PEM-1 Templates in PEM-1 BLOB Parameters

PEM-1 parameters (input or output) are combined to form PEM-1 Templates (Standard or Custom).

A PEM-1 Template is encapsulated in a BLOB as an XML document. An input parameter included in such XMLdocument is referred to as the PEM-1 Input BLOB Parameter. An output parameter included in such XML document is referred to as the PEM-1 Output BLOB Parameter. Each PEM-1 parameter representation in a PEM-1 Template is in the form conforming to the defined XML schema for the PEM-1 Template: it identifies the parameter name (see section 5.1.6) and is followed by the actual value of the PEM-1 Parameter. In case of two or more PEM-1 Parameters, the Template will include a concatenation of all parameters. The PEM-1 Parameters are all specified as part of the PEM-1 Template description (see section 5.1.6.and Appendix C for details), and therefore are known by the PEEM requestors and by the PEEM enabler implementation. The specified nature of the PEM-1 Parameters allows the PEEM enabler implementation, respectively the PEEM requestors to appropriately interpret the encoded PEM-1 Parameter representation during parsing of the PEM-1 Input BLOB Parameter, respectfully the PEM-1 Output BLOB Parameter. It is assumed that each PEEM enabler implementation will have access to a repository that defines all supported PEM-1 Templates, all supported PEM-1 Parameters, their corresponding data types, and optional allowable values, in accordance to the PEM-1 Templates definition. The implementation of such a repository, and how PEEM requestors and/or PEEM enabler implementation accesses the information in such repository is out-of-scope for the PEM-1TS. Similarly, how a Service Provider (SP) publishes and advertises the supported Standard and Custom PEM-1 Templates and Parameters is out-of-scope for the PEM-1 TS.

5.1.6 Encoding Scheme for PEM-1 Parameters in PEM-1 BLOB Parameters

PEM-1 specifies the use of plain XML text to represent the PEM-1 Templates (Input and/or Output) passed via the BLOB between a PEEM requestor and PEEM. No other binary encoding is specified in this release. The XML representation of the PEM-1 Templates is unique regardless of binding (SOAP or Diameter). Each parameter SHALL be represented with an XML tag, and an associated value. Appendix C specifies the PEM-1 Template schema, and how to build PEM-1 Templates using XML documents. The schema used for the PEM-1 Templates must conform to the schema made available by the Service Provider at deployment time.

5.2 Input / Output Standard PEM-1 Templates

Each PEM-1 Template (Standard or Custom) is composed of two or more PEM-1 parameters. Parameter names are represented through XML tags.

The parameter names in a Standard PEM-1 Template are defined and administered by OMA. The parameter names in a Custom PEM-1 Template are defined and administered by the Service Provider. The parameter names are associated with a value of pre-determined data type.

The PEM-1 Template SHALL have the following structure:

PEM-1 Template Header:	
Input/Output PEM-1 Parameter Name	**Input/Output Parameter Type**
templateID	STRING
templateVersion	STRING

Table 1: PEM-1 Template Header Structure

Where:

- templateID is the name of a PEM-1 parameter, representing a unique identifier of the template. It is mandatory for any PEM-1 template to include a templateID parameter.
    - For Standard PEM-1 Templates:
        - templateID values will be assigned by OMA using the process described in Appendix E.
    - For Custom PEM-1 Templates:
        - templateID values will be assigned by the Service Provider using their own process.
- templateVersion is the name of a PEM-1 parameter, representing the version of the template, and used to distinguish between multiple versions of the same template. It is mandatory for any PEM-1 template to include a templateVersion parameter.
    - For Standard PEM-1 Templates:
        - templateVersion values will be assigned by OMA using the process described in Appendix E
    - For Custom PEM-1 Templates:
        - templateVersion values will be assigned by the Service Provider using their own process.

A PEM-1 Template (Standard or Custom) always includes a PEM-1 Template Header and can have any number of additional I/O parameters. The mandatory parameters described in Table 1 are used to uniquely identify the remainder of the structure of a PEM-1 Template (i.e. identify the additional parameters).

5.2.1 Output Status Standard PEM-1 Template

The Output Status Template is the Standard PEM-1 Template that SHALL include any status codes returned by PEEM processing, or the policy processing . This section specifies how status codes can be returned as part of the PEM-1 interface output, regardless of protocol binding.

A PEEM implementation MUST support such a Standard PEM-1 Template (see Table 1).

Output Standard PEM-1 Template: **OutputStatus Template**	
Output PEM-1 Parameter Name	**Output Parameter Type**
PEM-1 Template Header	
statusCode	INTEGER
statusText	STRING

Table 2: OutputStatus Standard PEM-1 Template

Where:

- PEM-1 Template Header is as specified in section 5.2.1.
- statusCode is the name of a PEM-1 parameter, representing a final status result of the of the policy processing or an error identified by PEEM. This parameter is mandatory.
- statusCode values are assigned as defined in Appendix D, regardless of protocol binding.
- statusText is an optional parameter used to carry additional information of a specific Status code (see Appendix D).
- Any number of optional additional output parameters MAY be provided, as dictated by specific policy needs. The values associated to those parameter names can be of any supported type, as determined by the defined name. As a result of creating an OutputStatus Standard PEM-1Template for a specific enabler, additional parameters MAY be defined.

Note: The OutputStatus Standard PEM-1 template SHALL be passed as an XML document in an output BLOB. See specific PEM-1 template bindings for how a statusCode and the optional accompanying statusText may be exposed to the protocol using specific protocol methods, in addition to their inclusion in a binding-neutral fashion in the Output Status Standard PEM-1 Template.

The templateID to be used for the Output Standard PEM-1 template SHALL be OMA_PEEM_3_Output and the templateVersion for this template SHALL be V1.0.0, see also [OMNA].

5.2.2 Templates for identified policies

This section defines how policies specifically identified in a PEM-1 request are handled. Internal policy reference and external policy reference templates are defined in the next sections. In the case when an internal policy is identified in the PEM-1 request (see section 5.2.2.1 for details), a PEEM implementation MUST be able to process the identified policy, if such policy exists, without combining it in any way with other policies (i.e. the policy identified in the PEM-1 request is the only policy being processed with respect to the PEM-1 request). In the case when an external policy is identified in the PEM-1 request, a PEEM implementation that supports such an optional feature (see section 5.2.2.2 for details) MUST be able to process the identified policy, without combining it in any way with other policies (i.e. the policy identified in the PEM-1 request is the only policy being processed with respect to the PEM-1 request).

In addition, a PEEM implementation MAY support other ways to combine a policy identified via the PEM-1 request, with other policies. For example:

- process first any other policies that may apply,(other than the one indicated in PEM-1) and then process the policy identified via the PEM-1 request.
- vice versa.
- Combine the two policies, before processing them. When doing so, the way the two are combined should be deterministic, and made available (published) to PEEM requestors.

Supporting such other ways will allow dealing with any Service Provider's specific deployment choices. The ways in which the different options are configured, when available, is outside the scope of PEEM specifications.

5.2.2.1 Internal Policy reference Standard PEM-1 Template

This section specifies how a reference to an internal policy can be passed with a request through PEM-1. This template can be combined with any other Standard PEM-1 Template. A PEEM implementation uses the reference to identify a specific policy managed by PEM-2.

A PEEM implementation MUST support such a Standard PEM-1 Template.

The Input Policy reference Standard PEM-1 Template MAY be used as a preamble to an input BLOB or as part of an input/output Standard PEM-1 Template as specified in section 5.2.

In general, a PEEM implementation may use a combination of template parameters, in addition to other sources of information, to determine the policies to be evaluated. An explicit indication identifying the internal policy to be applied is realized using the following template:

Input Standard PEM-1 Template: InternalPolicyReference Template	
Input PEM-1 Parameter Name	**Input Parameter Type**
PEM-1 Template Header	
intPolicyID	URI

Table 3: Internal Policy Reference Standard PEM-1 Template

Where:

- For PEM-1 Template Header, see section 5.2.1.
- intPolicyID is the name of a PEM-1 parameter, representing a unique identifier of a policy internal to PEEM (i.e. a policy that can be managed using the PEM-2 interface). This parameter is mandatory.
    - intPolicyID values are assigned by the Service Provider.

The templateID to be used for the Input Standard PEM-1 template for internal policy reference SHALL be OMA_PEEM_1_Input_Internal and the templateVersion for this template SHALL be V1.0.0, see also [OMNA].

5.2.2.2 External Policy reference Standard PEM-1 Template

This section specifies how a reference to an external policy can be passed with a request through PEM-1. This template can be combined with any other Standard PEM-1 Template. A PEEM implementation then uses the external policy for the policy evaluation or evaluation and enforcement.

A PEEM implementation MAY optionally support such a Standard PEM-1 Template.

These PEM-1 parameters MAY be used as a preamble to an input BLOB or as part of an input/output Standard PEM-1 Template as specified in section 5.2.

A PEEM implementation may be configured to refuse input that include such a Standard PEM-1 Template, if the service provider or vendor wants to prevent passing policies as part of PEM-1 requests.

The use of an external policy is indicated by passing either a reference to the external policy, or passing the external policy itself by value, via a PEM-1 Parameter.

When passing an external policy by reference, an explicit indication identifying the external policy to be applied is realized using the following template:

Input Standard PEM-1 Template: ExternalPolicyByReference Template	
Input PEM-1 Parameter Name	**Input Parameter Type**
PEM-1 Template Header	
extPolicyID	URI

Table 4: External Policy By Reference Standard PEM-1 Template

Where:

- For PEM-1 Template Header, see section 5.2.1.
- extPolicyID is the name of a PEM-1 parameter, representing a policy that is being passed by a URI reference to PEEM. This parameter is mandatory.
    - extPolicyID values are URIs provided by the PEEM requestor.

When passing an external policy by value, an explicit indication identifying the external policy to be applied is realized using the following template:

Input Standard PEM-1 Template: ExternalPolicyByValue Template	
Input PEM-1 Parameter Name	**Input Parameter Type**
PEM-1 Template Header	
extPolicyVAL	STRING

Table 5: External Policy By Value Standard PEM-1 Template

Where:

- For PEM-1 Template Header,(see section 5.2.1.
- extPolicyVAL is the name of a PEM-1 parameter, representing a policy external to PEEM that is being passed by value to PEEM. This parameter is mandatory.
    - extPolicyVAL values are binary strings provided by the PEEM requestor.

The templateID to be used for the Input Standard PEM-1 template for external policy reference SHALL be OMA_PEEM_2_Input_External and the templateVersion for this template SHALL be V1.0.0, see also [OMNA].

5.3 I/O parameters

PEEM PEM-1 templates contain different combinations of parameters specified in this document, or added by other OMA WGs, vendors or Service Providers. In PEEM V1.0, the only specified input/output parameters are those described in the Standard PEM-1 Templates (section 5.2). Examples of additional possible parameters are given informatively in Appendix G.

5.3.1 PEM-1 parameters data types

The PEM-1 parameter data types include basic data types, selected derived data types (e.g. URI data type) and some complex data types. These data types SHALL be supported by any protocol bindings. Additional data types may be derived from these data types, on a need basis.

The table below represents the supported data types:

PEM-1 Data Types	Description
int	4 byte signed: -2147483648 to 2147483647
float	Floating-point number, 3.4e +/- 38 (7 digits)
array	Arrays (lists) of objects of a given type (e.g. arrays if integers, or characters, or floats).
function	A type that returns object of a given type.
struct	A complex type that contains a sequence of objects of different types.
string	A sequence (array) of characters
bool	A type that can only take the values TRUE or FALSE
URI	A type derived from string, with a well-specified structure as per [RFC 2396]

Table 6: PEM-1 parameters data types

5.4 PEM-1 Template Bindings

PEEM enabler implementations SHALL offer at least one of the following bindings for the PEM-1 Interface:

- Diameter
- SOAP

Other bindings are not precluded, but are not described in this specification.

5.4.1 PEM-1 Diameter binding

This section defines a transport protocol based on Diameter, to be used for requests/responses for OMA PEEM policy processing.

The provisions of this section are applicable between ANY resource and a PEEM enabler implementation, when using the Diameter binding. A Diameter PEEM client is a resource that uses the messages of the Diameter Policy Processing application to send a request with policy data for processing to a Diameter PEEM server. A Diameter PEEM server is a resource that uses the Diameter Policy Processing application to respond to a request for policy processing with an answer that may include policy data. In such case the PEEM enabler implementation acts as a Diameter PEEM server and resources sending policy processing requests to a PEEM enabler implementation act as Diameter PEEM clients.

Whenever it is possible this section specifies the requirements for this protocol by reference to specifications produced by the IETF within the scope of Diameter, and/or 3GPP within the scope of 3GPP Diameter applications. Where this is not possible, extensions to Diameter are defined within this section.

5.4.1.1 General

The Diameter Base Protocol as specified in IETF RFC 3588 [RFC 3588] SHALL apply except as modified by the defined support of the methods and the defined support of the commands and AVPs, result and event codes specified in clause 5.4.1.3 of this specification. Unless otherwise specified, the procedures (including error handling and unrecognized information handling) are unmodified. Use of the Diameter base application is detailed in clause 5.4.1.2 of this specification, and is informatively modelled after the 3GPP application Sh [**3GPP TS 29.329**], which in turn relies on 3GPP application Cx [**3GPP TS 29.229**].

5.4.1.2 Use of Diameter base application

With the clarifications listed in the following sub-clauses the Diameter Base Protocol defined by IETF RFC 3588 [RFC 3588] SHALL apply.

5.4.1.2.1 Securing Diameter Messages

This application does not introduce any new security measures. Securing Diameter messages SHALL conform to section 2.2 of IETF RFC 3588 [RFC 3588].

5.4.1.2.2 Accounting functionality

Accounting functionality (Accounting Session State Machine, related command codes and AVPs) is not used on the PEM-1 interface.

5.4.1.2.3 Use of sessions

Between a Diameter PEEM client and a Diameter PEEM server, Diameter sessions are implicitly terminated. An implicitly terminated session is one for which the server does not maintain state information. The client does not need to send any re-authorization or session termination requests to the server.

The Diameter base protocol includes the Auth-Session-State AVP as the mechanism for the implementation of implicitly terminated sessions.

The client (server) SHALL include in its requests (responses) the Auth-Session-State AVP set to the value NO_STATE_MAINTAINED (1), as described in IETF RFC 3588 [RFC 3588]. As a consequence, the server does not maintain any state information about this session and the client does not need to send any session termination request. Neither the Authorization-Lifetime AVP nor the Session-Timeout AVP SHALL be present in requests or responses.

5.4.1.2.4 Transport protocol

Diameter messages for the Policy Processing application SHALL use the mandated transport protocols specified in section 2.0 of IETF RFC 3588 [RFC 3588]. The Diameter server (PEEM implementation) SHALL support both TCP (IETF RFC 793 [RFC 793]) and SCTP (IETF RFC 2960 [RFC 2960]). A Diameter client (a PEEM requestor) MAY use either TCP or SCTP. When using SCTP, the new SCTP checksum method specified in RFC 3309 [RFC 3309] SHALL be used.

5.4.1.2.5 Routing considerations

This clause specifies the use of the Diameter routing AVPs Destination-Realm and Destination-Host. This application supports the routing mechanism specified in section 2 of IETF RFC 3588 [RFC 3588], and does not introduce any changes. In particular, if a PEEM requestor knows the specific address/name of the PEEM enabler implementation for a certain request, both the Destination-Realm and Destination-Host AVPs SHALL be present in the request. Otherwise, only the Destination-Realm AVP SHALL be present and the command SHALL be routed to the next Diameter node, based on the Diameter routing table in the client. Once the redirector function has returned the address of the destination PEEM enabler implementation (using Redirect-Host AVP), the redirected request to the PEEM enabler implementation SHALL include both Destination-Realm and Destination-Host AVPs. Consequently, the Destination-Host AVP is declared as optional in the ABNF [RFC 4234] for all requests initiated by a PEEM requestor. Host AVP is declared as mandatory in the ABNF [RFC 4234] for all requests initiated by the PEEM enabler implementation.

Destination-Realm AVP is declared as mandatory in the ABNF [RFC 4234] for all requests.

5.4.1.2.6 Advertising Application Support

A Diameter PEEM server SHALL advertise support of the Diameter Policy Processing application by including the value of the application identifier in the Auth-Application-Id AVP within the Vendor-Specific-Application-Id grouped AVP of the Capabilities-Exchange-Request and Capabilities-Exchange-Answer commands.

The vendor identifier value SHALL be included in the Supported-Vendor-Id AVP of the Capabilities-Exchange-Request and Capabilities-Exchange-Answer commands, and in the Vendor-Id AVP within the Vendor-Specific-Application-Id grouped AVP of the Capabilities-Exchange-Request and Capabilities-Exchange-Answer commands.

Note: The Vendor-Id AVP included in Capabilities-Exchange-Request and Capabilities-Exchange-Answer commands that is not included in the Vendor-Specific-Application-Id AVPs as described above SHALL indicate the manufacturer of the Diameter node as per IETF RFC 3588 [RFC 3588].

5.4.1.3 Diameter Policy Processing application

This clause specifies a Diameter application for Policy Processing invocation.

The Diameter Policy Processing application is defined as an IETF vendor specific Diameter application, where the vendor is the Open Mobile Alliance (OMA). The vendor identifier assigned by IANA to OMA (http://www.iana.org/assignments/enterprise-numbers) is **30079** (the IANA registered OMA Private Enterprise Number).

The Diameter application identifier assigned to the Policy Processing application is **16777243** (registered by IANA for Policy Processing application, PEEM 1.0).

5.4.1.3.1 Command-Code Values

This section defines Command-Code values for the Diameter Policy Processing application.

Every command is defined by means of the ABNF [RFC 4234] syntax, according to the rules in IETF RFC 3588 [RFC 3588]. Whenever the definition and use of an AVP is not specified in this document, and no reference is made to another specification, what is stated in IETF RFC 3588 [RFC 3588] SHALL apply.

The Command Code for the Diameter Policy Processing application is taken from the range allocated by IANA. For these commands, the Application-ID field SHALL be set to **16777243** (application identifier of the Diameter Policy Processing application, allocated by IANA) [RFC 5224].

The following Command Codes are defined in this specification:

Command-Name	Abbreviation	Code	Section
Policy-Data-Request	PDR	314	5.4.1.3.1.1
Policy-Data-Answer	PDA	314	5.4.1.3.1.2

Table 7: Command-Code values

5.4.1.3.1.1 Policy-Data-Request (PDR) Command

The Policy-Data-Request (PDR), indicated by the Command-Code field set to the value indicated in Table 7, section 5.4.1.3.1and the ‘R’ bit set in the Command Flags field, is sent by a Diameter PEEM client to a Diameter PEEM server in order to request policy data processing.

Message Format

< Policy-Data-Request> ::= < Diameter Header: 314, REQ, PXY, 16777243 >

< Session-Id >

{ 16777243 }

{ Auth-Session-State }

{ Origin-Host }

{ Origin-Realm }

[Destination-Host]

{ Destination-Realm }

{ Policy-Data }

*[Proxy-Info]

*[Route-Record]

*[AVP]

The AVPs indicated in **bold** represent new AVPs defined for this application; the other ones represent AVPs defined and supported by the Diameter base application. In general, Policy-Data is a container for all policy input parameters. The Policy-Data AVP does not encapsulate Diameter base protocol AVPs. Those AVPs are passed in the request as defined by the Diameter base protocol in IETF RFC 3588 [RFC 3588].

The entity acting as the Diameter PEEM server needs to be able to interpret the content of the Policy-Data AVP, according to the PEEM specification and/or the published custom specifications added by the Service Provider that deploys PEEM (see PEM-1 TS section 5.4.1.3.3).

5.4.1.3.1.2 Policy-Data-Answer (PDA) Command

The Policy-Data-Answer (PDA), indicated by the Command-Code field set to the value indicated in Table 7, section 5.4.1.3.1 and the 'R' bit cleared in the Command Flags field, is always sent back to the Diameter PEEM client by a Diameter PEEM server in response to the Policy-Data-Request command. The policy processing determines the content of the Policy-Data AVP.

Message Format

< Policy-Data-Answer > ::= < Diameter Header: 314, PXY, 16777243 >

< Session-Id >

{ 16777243 }

[Result-Code]

[Experimental-Result]

{ Auth-Session-State }

{ Origin-Host }

{ Origin-Realm }

{ Policy-Data }

*[Failed-AVP]

*[Proxy-Info]

*[Route-Record]

[Error-Message]

*[AVP]

The parameters indicated in **bold** represent new parameters defined for this application; the other ones represent parameters defined and supported by the Diameter base application. In general, Policy-Data is a container for all policy output parameters. In order to conform to the neutrally defined PEM-1 interface specification, the Policy-Data AVP for an answer SHALL always include the PEEM specific and/or policy processing specific status code and MAY include an optional status text, as well additional output results. The Policy-Data AVP does not encapsulate Diameter base protocol AVPs. Those AVPs are passed in the answer as defined by the Diameter base protocol in IETF RFC 3588 [RFC 3588]. In addition, the PEEM and/or policy processing status code and the accompanying optional status text are being exposed to the protocol using respectively the Experimental-Result AVP and the Error-Message AVP (see section 5.4.1.3.2 for details).

5.4.1.3.2 Mapping PEEM Status Codes to Diameter

This section defines how to expose status codes to the protocol. PEEM Diameter application uses Result-Code AVP, Experimental-Result AVP and Error-Message AVP.

The PEEM Diameter Application answer MUST use either a Result-Code AVP or an Experimental-Result AVP.

Result-Code AVP SHALL be used only to pass IETF registered status codes, as per RFC 3588. This specification does not define any new IETF registered status codes.

All other status codes that are PEEM specific status codes (success or failure of either PEEM or the result of a policy processing) SHALL be exposed using methods dictated by RFC 3588 for Vendor-specific Applications, using the Experimental-Result-Code AVP of the Experimental-Result grouped AVP, and MAY be accompanied by any optional status text using an optional Error-Message AVP. The Error-Message AVP value is not useful in real-time, but may be useful for other reasons (e.g. logging, or additional clarification).

All PEEM and/or policy processing status codes and all optional corresponding status texts are defined in Appendix D, in a joint format regardless of binding.

5.4.1.3.3 AVPs

The following table describes the Diameter AVPs defined for the Diameter PEEM application, their AVP Code values, types, possible flag values and whether the AVP may or not be encrypted.

				AVP Flag rules				
Attribute Name	**AVP Code**	**Section defined**	**Value Type**	**Must**	**May**	**Should not**	**Must not**	**May Encrypt**
Policy-Data	1	This document (this row), section 5.4.1.3.3	UTF8String	M, V				No. The Policy-Data is a container for all policy data parameters (input, output or used in exchanges with other resources) and they are encoded in a PEEM specified manner (see 5.1.6 for details)
NOTE 1: The AVP header bit denoted as ‘M’, indicates whether support of the AVP is required. The AVP header bit denoted as ‘V’, indicates whether the optional Vendor-ID field is present in the AVP header.								

Table 8: Diameter Policy Processing application AVPs

The Policy-Data AVP is of type UTF8String. This AVP (defined in the Vendor-Id namespace) is a container that can be used for exchanging:

1. Policy input parameters grouped in an XML document forwarded in the policy data request (PDR) by a requestor (acting as a Diameter PEEM client) to PEEM (acting as a Diameter PEEM server).
2. Policy output parameters grouped in an XML document sent in the policy data answer (PDA) by PEEM (acting as a Diameter PEEM server) back to the requestor (Diameter PEEM client), as a response to 1, above.

The parameters described above are are defined as an XML document, which is passed as plain text in an UTF8 string (Policy-Data AVP). The specific parameters contained in the XML document represented in the Policy-Data AVP are dictated by the policy and are either PEEM Standard Parameters or PEEM Custom Parameters, published by the PEEM deployer.

5.4.1.3.4 Special Requirements

5.4.1.3.4.1. Version Control

The following table SHALL apply to the Diameter Policy Processing application; the column Application identifier lists the used application identifiers used in OMA for this application.

Application identifier	First applied
16777243	OMA PEEM V1.0

Table 9: Application identifiers used in PEM-1

New functionality beyond the OMA PEEM V1.0 release SHALL be introduced by post-V1.0 versions of this specification to the Diameter applications as follows:

1. If possible, the new functionality SHALL be defined optional.
2. If backwards incompatible changes can not be avoided, the new functionality should be introduced as a feature, see 5.4.1.3.4.1.2.
3. If the change would be backwards incompatible even as if it was defined as a feature, a new version of the interface SHALL be created by changing the application identifier of the Diameter application, see 5.7.4.3.4.1.2.

5.4.1.3.4.2. New Feature

The base functionality for the Diameter Policy Processing application interface is the OMA PEEM V1.0 standard and a feature is an extension to that functionality. A feature is a functional entity that has a significant meaning to the operation of a Diameter application i.e. a single new parameter without a substantial meaning to the functionality of the Diameter endpoints should not be defined to be a new feature. If the support for a feature is defined mandatory in a post-V1.0 version of this specification, the feature concept enables interworking between Diameter endpoints regardless of whether they support all, some or none of the features of the application. Features should be defined so that they are independent from one another.

The content of a feature SHALL be defined as a part of the specification of the affected application messages. If new AVPs are added to the commands because of the new feature, the new AVPs SHALL have the ‘M’ bit cleared and the AVP SHALL not be defined mandatory in the command ABNF [RFC 4234]. The support for a feature may be defined to be mandatory behaviour of a node.

5.4.1.3.4.3. Changing the version of the interface

The version of an interface SHALL be changed by a future version of this specification only if there is no technically feasible means to avoid backwards incompatible changes to the Diameter application, i.e. to the current version of the interface. However, if the incompatible changes can be capsulated within a feature, there is no need to change the version of the interface. The versioning of an interface SHALL be implemented by assigning a new application identifier for the interface. This procedure is in line with the Diameter base protocol (see IETF RFC 3588 [RFC 3588]) which defines that if an

incompatible change is made to a Diameter application, a new application identifier SHALL be assigned for the Diameter application.

5.4.2 PEM-1 SOAP binding

This section defines a messaging protocol based on SOAP, to be used when invoking PEEM policy processing.

The provisions of this section are applicable between ANY resource and a PEEM enabler implementation, when using a SOAP binding.

SOAP messages will be passed from one SOAP processor to another using HTTP 1.1 as the protocol binding. A SOAP method is an HTTP request/response that complies with the SOAP1.1 encoding rules.

HTTP implicitly correlates its request message with its response message; therefore, an application using this binding can chose to infer a correlation between a SOAP message sent in the body of a HTTP request message and a SOAP message returned in the HTTP response. Similarly, HTTP identifies the server endpoint via a URI [URI], which can also serve as the identification of a SOAP processor at the node.

5.4.2.1 General (Web Services)

Only the key technologies in relation to PEEM SOAP binding are considered here. These comply with OWSER [OWSER].

SOAP messages over the PEM-1 interface SHALL make use of HTTP 1.1 IETF RFC 2616 [HTTP 1.1].

SOAP [SOAP 1.1] message represents the information needed to invoke a service or reflect the results of a service invocation, and contains the information specified in the service interface definition.

SOAP 1.1 is a standard, extensible, framework for packaging and exchanging XML messages, a convenient mechanism for referencing capabilities (typically by use of headers).

[SOAP 1.1] Part 1 defines an XML-based messaging framework: a processing model and an extensibility model.

[SOAP 1.1] Part 2 defines three optional components: a set of encoding rules for expressing instances of application-defined data types, a convention for representing remote procedure calls (RPC) and responses, and a set of rules for using SOAP with HTTP 1.1

WSDL 1.1 is a language for describing Web services.

WSDL describes Web services starting with the messages that are exchanged between the requestor and provider agents. The messages themselves are described abstractly and then bound to a concrete network protocol and message format.

With the clarifications listed in the following sub-clauses the SOAP Protocol defined by W3C in SOAP V1.1 SHALL apply.

5.4.2.1.1 Securing SOAP Messages

For secure transport of SOAP messages, see [WS Security 1.0].

Web Services Security provides end-to-end message-level security for web services through an implementation of the WS-security standard. WS-Security defines a mechanism for adding three levels of security to SOAP messages:

- Authentication tokens. WS-Security authentication tokens [USER NAME TOKEN PROFILE 1.0] let the client provide a user name and password or X.509 certificate [X.509 TOKEN PROFILE 1.0] for the purpose of authentication headers.

- XML encryption. WS-Security's use of W3C's XML encryption standard enables the XML body or portion of it to be encrypted to ensure message confidentiality.

- XML digital signatures. WS-Security's use of W3C's XML digital signatures lets the message be digitally signed to ensure message integrity. The signature is based on the content of the message itself (by applying the hash function and public key), so if the message is altered en route, the signature becomes invalid.

5.4.2.1.2 Routing considerations

This clause specifies the routing considerations for the web services interface.

Every endpoint has an address associated with it, which is used to locate and identify the endpoint. This address consists primarily of a Uniform Resource Identifier (URI), which specifies the location of the endpoint.

The **URI** is used in conjunction with any **Headers** to define an endpoint's SOAP Address Filter. By default, this filter verifies that an incoming message has a **To** message header that matches the endpoint's URI and that all of the required endpoint headers are present in the message.

The WSDL of a deployed Web Service (also called *dynamic WSDL*) includes an <address> element that assigns an address (URI) to a particular Web Service port

5.4.2.2 Web Service for PEM-1 interface

This clause specifies a Web Service that allows a Web Services client to send an input request for policy processing to the Web Services Provider (PEEM), and allows the Web Services provider to respond with an output response, if dictated by the invoked policy. The input request contains parameters needed for the policy processing, and the output response contains parameters produced by the policy processing.

The PEM-1 interface protocol is defined as a Web Service as per:
xmlns:callable_policy=http://www.openmobilealliance.org/wsdl/PEM1/v1_0/service

5.4.2.2.1 Web Services Messages

This section defines the Web Services calls for policy evaluation request and response based on the SOAP [SOAP 1.1] protocol. The Service is specified in WSDL [WSDL1.1].

The policy evaluation interface makes it possible for an external application to evaluate a request containing a set of parameters. The Attributes are sent according to the defined Input Template used for the application. The attributes SHALL be sent as XML formatted data according to the Input Template XML Schema defined for the application. See Appendix C for details about the XML Schemas.

Every message is defined by means of the [SOAP 1.1] syntax, according to the rules in W3C.

The following messages are defined in this specification:

Message-Name	Section
evaluatePolicyRequest	5.4.2.2.1.1
evaluatePolicyResponse	5.4.2.2.1.2
policyResultRequest	5.4.2.2.1.4
policyResultResponse	5.4.2.2.1.5
denyPolicyResponseException	5.4.2.2.1.3
Error exceptions	5.4.2.2.1.6
SOAP Errors	5.4.2.2.1.7

Table 10: Messages

The WSDL is split in three files. One contains the request/response to PEEM [PEEM REQ WSDL], another contains the policy result call-back [PEEM RSP WSDL] and finally another one describes the WSDL faults including the synchronous response to a statusCode of DENY type (e.g. statusCode = 0x2401, DENY decision, or statusCode = 0x2402, DENY decision with additional results). Please see the WSDL files in [PEEM FAULT WSDL].

The [PEEM REQ WSDL] WSDL file contains the interface description for the policy enforcement point calling the PEEM for a policy evaluation using the *evaluatePolicyRequest* method as listed below. The faults are defined in the [PEEM FAULT WSDL] WSDL file.

The [PEEM RSP WSDL] WSDL file contain the definition of the asynchronous response, where PEEM uses the *policyResultRequest* as listed below to send the result of a policy request to a defined receiver. The *policyResultRequest* is listed below. The receiver SHALL return the *policyResultResponse* which is an empty acknowledge.

5.4.2.2.1.1. Web Services evaluatePolicyRequest message

The *evaluatePolicyRequest*, Message is sent by a Web Services client (PEEM requestor) to a PEEM server in order to request policy processing. The *evaluatePolicyRequest* WSDL is shown below see also [PEEM REQ WSDL].

```
<xsd:complexType name="evaluatePolicyRequest_type">
   <xsd:sequence>
      <xsd:element name="callbackUrl" type="xsd:string" minOccurs="0" maxOccurs="1" />
      <xsd:element name="timeStamp" type="xsd:dateTime" />

      <!--
         policyData contains an xml based document containing the additional policy data
         e.g. variables and their values.
        -->
      <xsd:element name="policyData" type="xsd:anyType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<message name="evaluatePolicyRequest">
   <part name="parameters" type="callable_policy_local_xsd:evaluatePolicyRequest_type" />
</message>
```

The policy evaluation interface makes it possible for an external application to evaluate a request containing a set of parameters. The parameters in the request includes a callable URL, a time stamp and a container for arbitrary additional data provided as name-value pairs.

- **Synchronous mode** is shown in Figure 1 and Figure 2 below. The result SHALL be sent in the *evaluatePolicyResponse* message if the statusCode are one of ALLOW (2101-2110) or SUCCESS (2701-2710), see Figure 1 for an example. In the case of a DENY (2401-2410) type status code the service SHALL return the *denyPolicyResponseException* as described in 5.4.2.2.1.3, see Figure 2 for an example.
- **Asynchronous mode** is shown in Figure 3 below. In the Asynchronous mode the final response SHALL be sent in a *policyResultRequest* Message as described in 5.4.2.2.1.5 below.

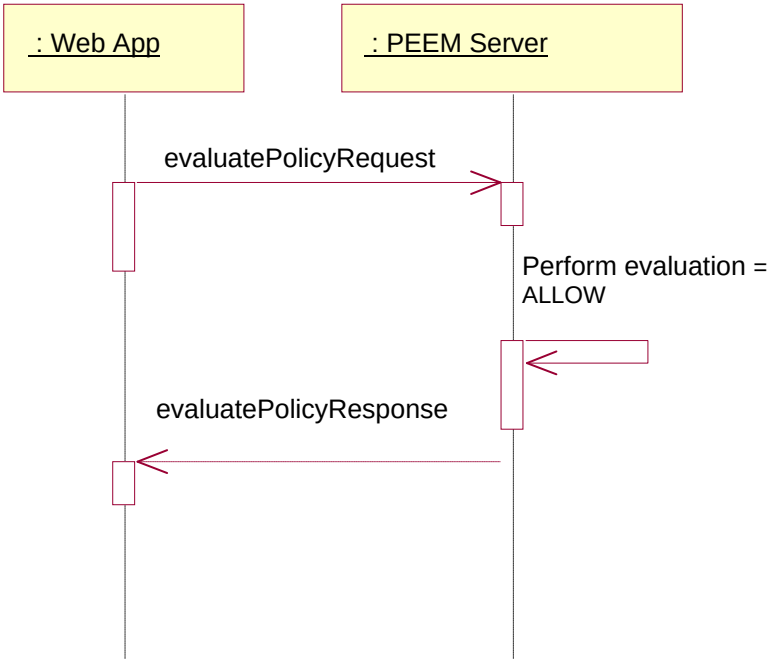


Figure 1: Synchronous evaluatePolicyResponse message to original destination, evaluate to ALLOW (2101-2110) or SUCCESS (2701-2710) type Status Code.

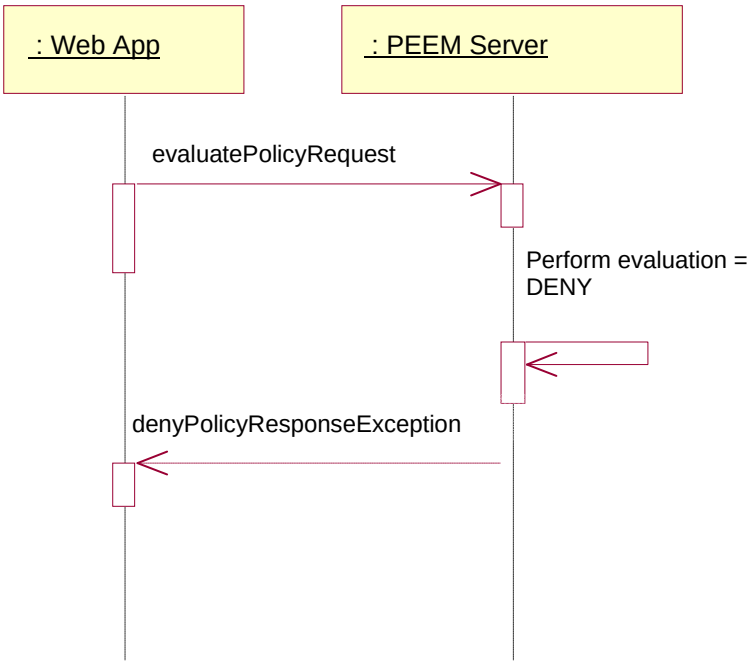


Figure 2: Synchronous evaluatePolicyResponse message to original destination, evaluate to DENY (2401-2410) type StatusCode

5.4.2.2.1.2. Web Services evaluatePolicyResponse Message

The *evaluatePolicyResponse* message, is returned by a PEEM server synchronously in response to the *evaluatePolicyRequest* message as shown in Figure 2 above. The *evaluatePolicyResponse* WSDL is shown below see also [PEEM REQ WSDL].

```
<xsd:complexType name="evaluatePolicyResponse_type">
   <xsd:sequence>
      <xsd:element name="correlator" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <!--
             StatusCode contains the result of the request, e.g. the following
             standard PEEM status codes.

                 2101 = ALLOW decision
                 2102 = ALLOW decision with additional results

                 2701 = SUCCESS (Policy processing was successful, but it did not
                                 require rendering an ALLOW or DENY)
                 2702 = SUCCESS with additional result

                 In case of Denial of a synchronous request the
                 denyPolicyResponseException is returned. In case of
                 any other Status code different exceptions are returned.
         -->
      <xsd:element name="statusCode" type="xsd:unsignedShort" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="statusText" type="xsd:string" minOccurs="0" maxOccurs="1"/>

      <!--
         policyData contains an xml based document containing the additional policy data
         e.g. variables and their values.
         -->
      <xsd:element name="policyData" type="xsd:anyType" minOccurs="0" maxOccurs="1"/>

   </xsd:sequence>
</xsd:complexType>

<message name="evaluatePolicyResponse">
   <part name="result" type="callable_policy_local_xsd:evaluatePolicyResponse_type" />
</message>
```

The evaluatePolicyResponse SHALL be returned:

- If the callbackURL in the evaluate policy request is not set indicating synchronous mode and the resulting statusCode are of ALLOW (2101-2110) or SUCCESS (2701-2710) type. The response SHALL be as defined for any PEEM protocol binding contain the statusCode, with an optional statusText both copied from the Output Status Template. The response as well contains a list of variables from the policy execution in the policyData parameter according to the used Policy Output Template. The correlator parameter SHALL not be returned.
- If the callbackURL in the evaluate policy request is set indicating asynchronous mode the *evaluatePolicyResponse* SHALL be returned without any statusCode, statusText or policyData. A PEEM Server unique correlation identifier SHALL be returned in the correlator parameter. The form of this correlation identifier is a vendor specific string that SHALL be unique for each request. The policy decision SHALL then later be sent asynchronously as described in 5.4.2.1.1.4 below.

5.4.2.2.1.3. Web Services denyPolicyResponseException

The *denyPolicyResponseException* message is returned by a server in the response to a synchronous *evaluateReponseRequest* message (I.e. a message without a callback URI) as shown in Figure 2 above, when the resulting

statusCode are 2401 (DENY decision) or 2402 (DENY decision with additional results). The statusText, copied from the Output Status Template, can optionally contain a text describing the reason for the denial of service. The Policy Data parameter can contain additional variables according to the Output Status Template. Any additional DENY type result codes, application or vendor specific SHALL be reported using the *denyPolicyResponseException* as well.

The *denyPolicyResponseException* WSDL is shown below see also[PEEM FAULT WSDL].

```
<xsd:complexType name="denyPolicyResponseException_type">
   <xsd:sequence>
      <!--
          evaluation Result is the verdict,
              StatusCode
                2401 = Policy Denies Service
                2402 = DENY decision with additional results

          StatusText contains additional information for the denial
         -->
       <xsd:element name="statusCode" type="xsd:unsignedShort" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="statusText" type="xsd:string" minOccurs="0" maxOccurs="1"/>

       <!--
          policyData contains an xml based document containing the additional policy data
          e.g. variables and their values.
          -->
       <xsd:element name="policyData" type="xsd:anyType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<wsdl:message name="denyPolicyResponseException">
   <wsdl:part name="denyPolicyResponseException" type="typens:denyPolicyResponseException_type"/>
</wsdl:message>
```

5.4.2.2.1.4. Web Services policyResultRequest Message

The *policyResultRequest* message is sent asynchronously by the PEEM server as a result of the Policy Evaluation initiated by the *evaluatePolicyRequest* message as shown in Figure 3. The policyResultRequest SHALL contain the same correlation identifier as was returned in the response to the *evaluatePolicyRequest*. The *policyResultRequest* SHALL contain the resulting policy data from the evaluation process and the resulting StatusCode and optional StatusText. The *policyResultRequest* SHALL be sent to the address included in the *callbackURL* defined in the initial *evaluatePolicyRequest*.

The *policyResultRequest* WSDL is shown below see also [PEEM RSP WSDL].

```
<xsd:complexType name="policyResultRequest">
   <xsd:sequence>
      <xsd:element name="correlator" type="xsd:string" />

      <!--
            Status Code is the verdict,
               StatusCode
                  2101 = ALLOW decision
                  2102 = ALLOW decision with additional results
```

```
                  2401 = DENY decision
                  2402 = DENY decision with additional results

                  2701 = SUCCESS (Policy processing was successful, but it did not require
                           rendering an ALLOW or DENY)
                  2702 = SUCCESS with additional result
         -->
         <xsd:element name="statusCode" type="xsd:unsignedShort" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="statusText" type="xsd:string" minOccurs="0" maxOccurs="1"/>

      <!--
         policyData contains an xml based document containing the additional policy data
         e.g. variables and their values.
        -->
      <xsd:element name="policyData" type="xsd:anyType" minOccurs="0" maxOccurs="1"/>

   </xsd:sequence>
</xsd:complexType>

<message name="policyResultRequest">
   <part name="parameters" type="callable_policy_resp_local_xsd:policyResultRequest" />
</message>
```

The *statusCode* and the optional *statusText*, wich are copied from the Output Status Template specify the result of the policy evaluation. The policyData parameter contains additional data as specified by the output status template in XML format, see Appendix C for details on the used XML Schema.

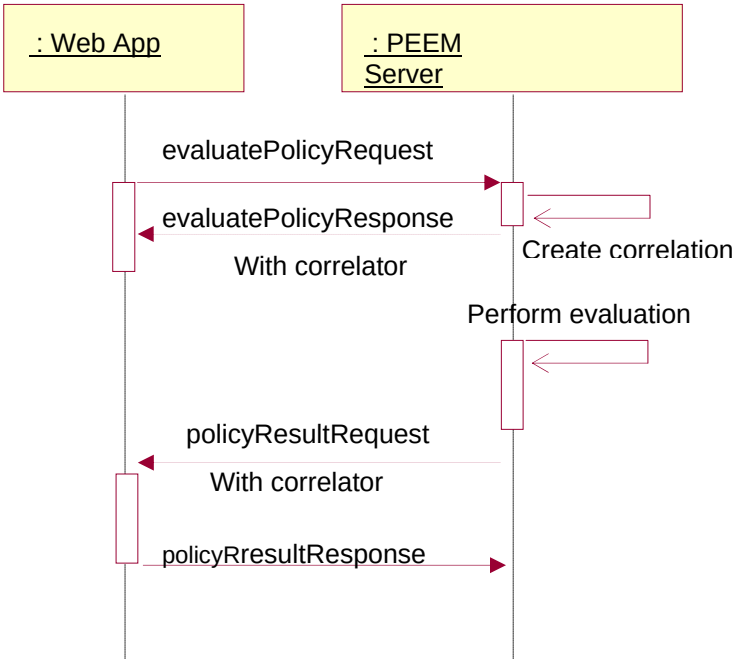


Figure 3: Asynchronous policyResultRequest message response to same or different destination

Note that in order for such a 3 way interaction (PEEM requestor, PEEM enabler implementation, PEEM response destination) to be supported by this specification, the PEEM response destination resource needs to fully support this specification, and the PEEM enabler implementation must be able to act as both Web services server and Web Services client. Finally, note that in this case, any interactions between PEEM and other resources than the PEEM requestor can be conducted using this specification (in other words, PEEM enabler implementation would act as a Web Services client, with the other resources acting as Web Services servers).

5.4.2.2.1.5. Web Services policyResultResponse Message (including responses to a different destination)

The *policyResultResponse* message, SHALL be returned immediately by the receiver of the *policyResultRequest* as shown in Figure 3. The *policyResultResponse* message is an empty message.

5.4.2.2.1.6. PEEM Web Services binding - Error Messages

All the ordinary Policy results/status codes are handled as described above as part of the *evaluatePolicyResponse* and *denyPolicyResponseException* or via the asynchronous *policyResultRequest* call. In the case of any other status code indicating errors like malformed XML data or similar the defined WSDL Fault structures is used as defined in [PEEM FAULT WSDL] called depending on the type of fault class that has occurred. The specification defines four classes:

- Informational Status that are handled using the *InformationlStatusException,*
- Protocol Errors Status errors that are handled using the *protocolErrorException,*
- Transient Failures Status errors that are handled using the transientFailureException *and*
- Permanent Failures Status errors that are handled using the permanentFailure*Exception*.

The *InformationlStatusException, protocolErrorException, transientFailureException* or *permanentFailureException* are all defined with three data items as described below.

The mapping of the error information SHALL be done as follows:

- The first data is the mandatory statusCode parameter carrying the uniquely defined status code, as defined in Appendix D below.
- The optional second parameter is the statusText containing the error description of the error, including placeholders (marked with %) for additional information. This form is consistent with the form for internationalization of messages used by many technologies (operating systems, programming environments, etc.). Use of this form enables translation of messages to different languages independent of program changes. This is well suited for Web Services messages, as a programming language is not defined. A proposal for the text of the PEEM standard errors is given in column 2 in the tables in Appendix D and in the SOAP binding specific errors [SOAP 1.1] and [SOAP 1.2].
- The third data item variable is a list of zero or more strings that represent the content to put in each placeholder defined in the message in the statusText parameter.

The standard list of status codes is specified in Appendix D. Additional errors can be added by implementations for other errors, according to Appendix D.6.

5.4.2.2.1.7. SOAP Error Message

In addition to the WSDL/Application errors defined in Appendix D, SOAP errors can be returned from the PEEM implementation due to lower layer protocol errors. These are following the standard SOAP and HTTP standards. See [SOAP 1.1], [SOAP 1.2] and [WSDL 1.1].

5.4.2.2.2 Version Control

The following table SHALL apply to the PEM-1 interface; the column Application identifier lists the used application identifiers on PEM-1 and OMA.

Target Name Space	First applied
http://www.openmobilealliance.org/schema/PEM1/v1_0	OMA PEEM V1.0

Table 11: WSDL Target Name Space used in PEM-1

Name space versioning will be used for new functionality beyond the OMA PEEM V1.0 release & SHALL be introduced by post-V1.0 versions of this specification to the SOAP applications as follows:

1. If possible, the new functionality SHALL be defined optional.
2. If backwards incompatible changes can not be avoided, the new functionality should be introduced as a feature, see 5.4.2.2.3.
3. If the change would be backwards incompatible even as if it was defined as a feature, a new version of the interface SHALL be created by changing the application identifier of the Web Services application, see 5.4.2.2.4.

5.4.2.2.3 New Feature

The base functionality for the PEM-1 interface is the OMA PEEM V1.0 standard and a feature is an extension to that functionality.

5.4.2.2.4 Changing the version of the interface

The version of an interface SHALL be supported by adding a new namespace.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-PEEM_PEM1-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. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [IOPPROC].

In the following, the Server is the PEEM implementation and the Client is the Policy Evaluation Requestor (not part of the PEEM enabler).

B.1 SCR for PEM-1 Client Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_BINDING-C-001-M	Support of bindings	Section 5.4	PEEM-PEM1_DIAMETER-C-001-O OR PEEM-PEM1_SOAP-C-001-O OR PEEM-PEM1_BLOB-C-001-O

B.2 SCR for PEM-1 Diameter Client Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_DIAMETER-C-001-O	Diameter protocol binding	Section 5.4.1	PEEM-PEM1_DIAMETER-C-002-O AND PEEM-PEM1_DIAMETER-C-003-O AND PEEM-PEM1_DIAMETER-C-005-O AND PEEM-PEM1_DIAMETER-C-006-O AND PEEM-PEM1_DIAMETER-C-007-O AND PEEM-PEM1_DIAMETER-C-012-O
PEEM-PEM1_DIAMETER-C-002-O	Use of Diameter base protocol	Section 5.4.1.1	
PEEM-PEM1_DIAMETER-C-003-O	Use of Diameter base application	Section 5.4.1.2	
PEEM-PEM1_DIAMETER-C-004-O	Securing Diameter messages	Section 5.4.1.2.1	
PEEM-PEM1_DIAMETER-C-005-O	Used transport protocol	Section 5.4.1.2.4	
PEEM-PEM1_DIAMETER-C-006-O	Use of Command-Code values	Section 5.4.1.3.1	
PEEM-PEM1_DIAMETER-C-007-O	Synchrounous service	Section 5.4.1.3.1	PEEM-PEM1_DIAMETER-C-008-O AND PEEM-PEM1_DIAMETER-C-009-O AND PEEM-PEM1_DIAMETER-C-010-O AND PEEM-PEM1_DIAMETER-C-011-O

Item	Function	Reference	Requirement
PEEM-PEM1_DIAMETER-C-008-O	Send Policy-Data-Request command	Section 5.4.1.3.1.1	
PEEM-PEM1_DIAMETER-C-009-O	Receive Policy-Data-Response command	Section 5.4.1.3.1.2	
PEEM-PEM1_DIAMETER-C-010-O	Retrieveing PEEM status codes from Diameter	Section 5.4.1.3.2	PEEM-PEM1_DIAMETER-C-009-O
PEEM-PEM1_DIAMETER-C-011-O	Retrieving PEEM policy data from Diameter	Section 5.4.1.3.1.2	PEEM-PEM1_DIAMETER-C-009-O
PEEM-PEM1_DIAMETER-C-012-O	Version control of Diameter	Section 5.4.1.3.4.1	

B.3 SCR for PEM-1 SOAP Client Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_SOAP-C-001-O	SOAP protocol binding	Section 5.4.2	PEEM-PEM1_SOAP-C-002-O AND PEEM-PEM1_SOAP-C-004-O AND PEEM-PEM1_SOAP-C-005-O AND PEEM-PEM1_SOAP-C-006-O AND PEEM-PEM1_SOAP-C-014-O AND PEEM-PEM1_SOAP-C-015-O AND PEEM-PEM1_SOAP-C-016-O
PEEM-PEM1_SOAP-C-002-O	Use of SOAP and Web Services	Section 5.4.2.1	
PEEM-PEM1_SOAP-C-003-O	Use of WS-Security for securing SOAP binding	Section 5.4.2.1.1	
PEEM-PEM1_SOAP-C-004-O	Use of Web Services name space	Section 5.4.2.2	
PEEM-PEM1_SOAP-C-005-O	Synchrounous service	Section 5.4.2.2.1.1	PEEM-PEM1_SOAP-C-007-O AND PEEM-PEM1_SOAP-C-008-O AND PEEM-PEM1_SOAP-C-011-O AND PEEM-PEM1_SOAP-C-012-O AND PEEM-PEM1_SOAP-C-013-O
PEEM-PEM1_SOAP-C-006-O	Asynchrounous service	Section 5.4.2.2.1.1	PEEM-PEM1_SOAP-C-007-O AND PEEM-PEM1_SOAP-C-008-O AND PEEM-PEM1_SOAP-C-009-O AND PEEM-PEM1_SOAP-C-010-O AND PEEM-PEM1_SOAP-C-011-O AND PEEM-PEM1_SOAP-C-012-O AND PEEM-PEM1_SOAP-C-013-O
PEEM-PEM1_SOAP-C-007-O	Send evaluatePolicyRequest message	Section 5.4.2.2.1.1	
PEEM-PEM1_SOAP-C-008-O	Receive evaluatePolicyResponse message	Section 5.4.2.2.1.2	

Item	Function	Reference	Requirement
PEEM-PEM1_SOAP-C-009-O	Receive policyResultRequest message	Section 5.4.2.2.1.3	
PEEM-PEM1_SOAP-C-010-O	Send policyResultResponse message	Section 5.4.2.2.1.4	
PEEM-PEM1_SOAP-C-011-O	Receive denyPolicyResponseException exception	Section 5.4.2.2.1.5	
PEEM-PEM1_SOAP-C-012-O	Receive Error exceptions	Section 5.4.2.2.1.6	
PEEM-PEM1_SOAP-C-013-O	SOAP Errors	Section 5.4.2.2.1.7	
PEEM-PEM1_SOAP-C-014-O	Retrieving PEEM status codes from SOAP	Section 5.4.2.2.1.2, Section 5.4.2.2.1.3, Section 5.4.2.2.1.4, Section 5.4.2.2.1.6,	PEEM-PEM1_SOAP-C-008-O OR PEEM-PEM1_SOAP-C-009-O OR PEEM-PEM1_SOAP-C-011-O OR PEEM-PEM1_SOAP-C-012-O
PEEM-PEM1_SOAP-C-015-O	Retrieving PEEM policy data from SOAP	Section 5.4.2.2.1.2, Section 5.4.2.2.1.3, Section 5.4.2.2.1.4	PEEM-PEM1_SOAP-C-008-O OR PEEM-PEM1_SOAP-C-009-O OR PEEM-PEM1_SOAP-C-011-O
PEEM-PEM1_SOAP-C-016-O	Version control of SOAP	Section 5.4.2.2.2	

B.4 SCR for PEM-1 BLOB

Item	Function	Reference	Requirement
PEEM-PEM1_BLOB-C-001-O	BLOB Interface	Section 5.1.1	

B.5 SCR for PEM-1 Client

Item	Function	Reference	Requirement
PEEM-PEM1-C-001-M	BLOB Interface encapsulates the Templates	Section 5.1.1, Section 5.1.5	PEEM-PEM1-C-005-M
PEEM-PEM1-C-002-M	PEM-1 interface data defined by the Templates	Section 5.1.2	PEEM-PEM1-C-005-M
PEEM-PEM1-C-003-M	Need to support the Standard Template	Section 5.1.3, Section 5.2.1, Section 5.2.2	PEEM-PEM1-C-001-M AND PEEM-PEM1-C-004-M AND PEEM-PEM1-C-005-O
PEEM-PEM1-C-004-M	Support for Internal Policy Reference Standard PEM-1	Section 5.2.2.1	

Item	Function	Reference	Requirement
	template		
PEEM-PEM1-C-005-O	Support for External Policy Reference Standard PEM-1 template	Section 5.2.2.2	
PEEM-PEM1-C-006-O	Support for Custom Templates	Section 5.1.4	PEEM-PEM1-C-001-M
PEEM-PEM1-C-007-M	BLOB parameter encoded in XML	Section 5.1.6	
PEEM-PEM1-C-008-M	Allowed parameter types	Section 5.3.1	PEEM-PEM1-C-005-M
PEEM-PEM1-C-009-M	Protocol binding	Section 5.4	PEEM-PEM1_BINDING-C-001-M

B.6 SCR for PEM-1 Server Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_BINDING-S-001-M	Support of bindings	Section 5.4	PEEM-PEM1_DIAMETER-S-001-O OR PEEM-PEM1_SOAP-S-001-O OR PEEM-PEM1_BLOB-S-001-O

B.7 SCR for PEM-1 Diameter Server Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_DIAMETER-S-001-O	Diameter protocol binding	Section 5.4.1	PEEM-PEM1_DIAMETER-S-002-O AND PEEM-PEM1_DIAMETER-S-003-O AND PEEM-PEM1_DIAMETER-S-005-O AND PEEM-PEM1_DIAMETER-S-006-O AND PEEM-PEM1_DIAMETER-S-007-O AND PEEM-PEM1_DIAMETER-S-008-O AND PEEM-PEM1_DIAMETER-S-009-O AND PEEM-PEM1_DIAMETER-S-014-O
PEEM-PEM1_DIAMETER-S-002-O	Use of Diameter base protocol	Section 5.4.1.1	
PEEM-PEM1_DIAMETER-S-003-O	Use of Diameter base application	Section 5.4.1.2	
PEEM-PEM1_DIAMETER-S-004-O	Securing Diameter messages	Section 5.4.1.2.1	

Item	Function	Reference	Requirement
PEEM-PEM1_DIAMETER-S-005-O	Used transport protocol	Section 5.4.1.2.4	
PEEM-PEM1_DIAMETER-S-006-O	Routing of Diamater messages	Section 5.4.1.2.5	
PEEM-PEM1_DIAMETER-S-007-O	Advertising application support	Section 5.4.1.2.6	
PEEM-PEM1_DIAMETER-S-008-O	Use of Command-Code values	Section 5.4.1.3.1	
PEEM-PEM1_DIAMETER-S-009-O	Synchrounous service	Section 5.4.1.3.1	PEEM-PEM1_DIAMETER-S-010-O AND PEEM-PEM1_DIAMETER-S-011-O AND PEEM-PEM1_DIAMETER-S-012-O AND PEEM-PEM1_DIAMETER-S-013-O
PEEM-PEM1_DIAMETER-S-010-O	Policy-Data-Request command	Section 5.4.1.3.1.1	
PEEM-PEM1_DIAMETER-S-011-O	Policy-Data-Response command	Section 5.4.1.3.1.2	PEEM-PEM1_DIAMETER-S-012-O AND PEEM-PEM1_DIAMETER-S-013-O
PEEM-PEM1_DIAMETER-S-012-O	Mapping PEEM status codes to Diameter	Section 5.4.1.3.2	
PEEM-PEM1_DIAMETER-S-013-O	Mapping PEEM policy data to Diameter	Section 5.4.1.3.2	
PEEM-PEM1_DIAMETER-S-014-O	Version control of Diameter	Section 5.4.1.3.4.1	

B.8 SCR for PEM-1 SOAP Server Protocol Binding

Item	Function	Reference	Requirement
PEEM-PEM1_SOAP-S-001-O	SOAP protocol binding	Section 5.4.2	PEEM-PEM1_SOAP-S-002-O AND PEEM-PEM1_SOAP-S-004-O AND PEEM-PEM1_SOAP-S-005-O AND PEEM-PEM1_SOAP-S-006-O AND PEEM-PEM1_SOAP-S-015-O AND PEEM-PEM1_SOAP-S-016-O AND PEEM-PEM1_SOAP-S-017-O
PEEM-PEM1_SOAP-S-002-O	Use of SOAP and Web Services	Section 5.4.2.1	
PEEM-PEM1_SOAP-S-003-O	Use of WS-Security for securing SOAP binding	Section 5.4.2.1.1	
PEEM-PEM1_SOAP-	Routing of SOAP	Section	

Item	Function	Reference	Requirement
S-004-O	messages	5.4.2.1.2	
PEEM-PEM1_SOAP-S-005-O	Use of Web Services name space	Section 5.4.2.2	
PEEM-PEM1_SOAP-S-006-O	Synchrounous service	Section 5.4.2.2.1.1	PEEM-PEM1_SOAP-S-008-O AND PEEM-PEM1_SOAP-S-009-O AND PEEM-PEM1_SOAP-S-012-O AND PEEM-PEM1_SOAP-S-013-O AND PEEM-PEM1_SOAP-S-014-O
PEEM-PEM1_SOAP-S-007-O	Asynchrounous service	Section 5.4.2.2.1.1	PEEM-PEM1_SOAP-S-008-O AND PEEM-PEM1_SOAP-S-009-O AND PEEM-PEM1_SOAP-S-011-O AND PEEM-PEM1_SOAP-S-011-O AND PEEM-PEM1_SOAP-S-012-O AND PEEM-PEM1_SOAP-S-013-O AND PEEM-PEM1_SOAP-S-014-O
PEEM-PEM1_SOAP-S-008-O	evaluatePolicyRequest message	Section 5.4.2.2.1.1	
PEEM-PEM1_SOAP-S-009-O	evaluatePolicyResponse message	Section 5.4.2.2.1.2	
PEEM-PEM1_SOAP-S-010-O	policyResultRequest message	Section 5.4.2.2.1.3	
PEEM-PEM1_SOAP-S-011-O	policyResultResponse message	Section 5.4.2.2.1.4	
PEEM-PEM1_SOAP-S-012-O	denyPolicyResponseException exception	Section 5.4.2.2.1.5	
PEEM-PEM1_SOAP-S-013-O	Error exceptions	Section 5.4.2.2.1.6	
PEEM-PEM1_SOAP-S-014-O	SOAP Errors	Section 5.4.2.2.1.7	
PEEM-PEM1_SOAP-S-015-O	Mapping PEEM status codes to SOAP	Section 5.4.2.2.1	
PEEM-PEM1_SOAP-S-016-O	Mapping PEEM policy data to SOAP	Section 5.4.2.2.1	
PEEM-PEM1_SOAP-S-017-O	Version control of SOAP	Section 5.4.2.2.2	

B.9 SCR for PEM-1 BLOB

Item	Function	Reference	Requirement
PEEM-PEM1_BLOB-S-001-O	BLOB Interface	Section 5.1.1	

B.10 SCR for PEM-1 Server

Item	Function	Reference	Requirement
PEEM-PEM1-S-001-M	BLOB Interface encapsulates the Templates	Section 5.1.1, Section 5.1.5	PEEM-PEM1-S-005-M
PEEM-PEM1-S-002-M	PEM-1 interface data	Section 5.1.2	PEEM-PEM1-S-005-M

Item	Function	Reference	Requirement
	defined by the Templates		
PEEM-PEM1-S-003-M	Need to support the Standard Template	Section 5.1.3, Section 5.2.1, Section 5.2.2	PEEM-PEM1-S-001-M AND PEEM-PEM1-S-004-M AND PEEM-PEM1-S-005-O
PEEM-PEM1-S-004-M	Support for Internal Policy Reference Standard PEM-1 template	Section 5.2.2.1	
PEEM-PEM1-S-005-O	Support for External Policy Reference Standard PEM-1 template	Section 5.2.2.2	
PEEM-PEM1-S-006-O	Support for Custom Templates	Section 5.1.4	PEEM-PEM1-S-001-M
PEEM-PEM1-S-007-M	BLOB parameter encoded in XML	Section 5.1.6	
PEEM-PEM1-S-008-M	Allowed parameter types	Section 5.3.1	PEEM-PEM1-S-005-M
PEEM-PEM1-S-009-M	Protocol binding	Section 5.4	PEEM-PEM1_BINDING-S-001-M

Appendix C. Normative PEM-1 Template Bindings (Normative)

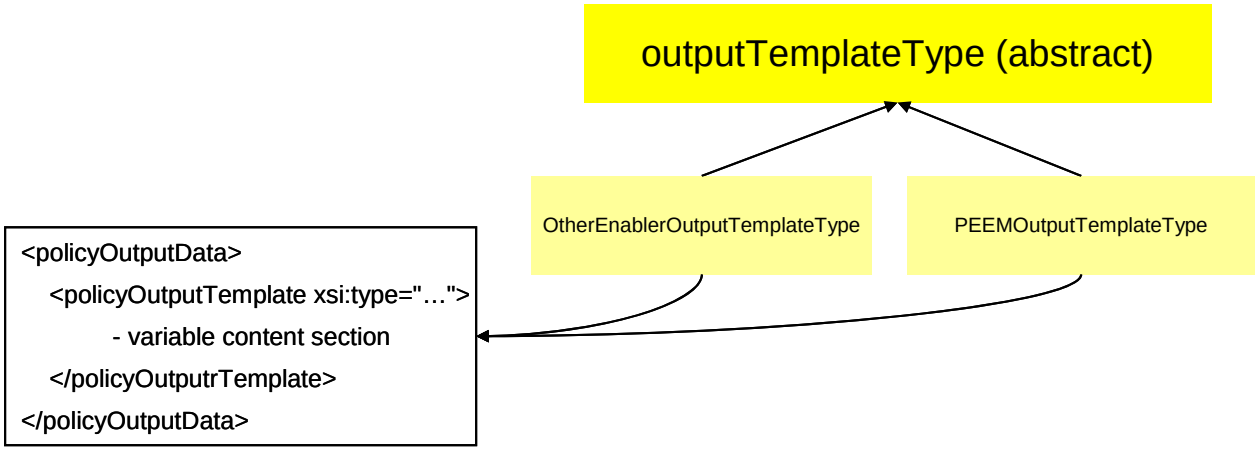
The PEM-1 input and output standard templates are defined using XSD Schemas binding. The BLOBs in the PEM-1 protocol bindings contains the resulting XML data.

The standard XSD Schemas are based on an XSD mechanism for extensions as described below. See also the PEEM standard Input policy data template XSD [PEEM INPUT TEMPLATE XSD] and the PEEM standard Output policy data template XSD [PEEM OUTPUT TEMPLATE XSD] for details.

C.1 Implementing variable content containers using an abstract type and type substitution

There are three XML Schema concepts that are used implementing this method:

- A complex type to be declared abstract.
- An element declared to be of an abstract type cannot have its type instantiated in instance documents (that is, the element can be instantiated, but its abstract content may not).
- In instance documents an element with an abstract type must have its content substituted by content from a non-abstract (concrete) type which derives from the abstract type. This is called type substitution.



C.1.1 Implementation Input Template

Define the abstract base types (inputTemplateType and externalPolicyRulesTemplateType):

```
<xs:complexType name="inputTemplateType" abstract="true">
     <xs:sequence>
          <xs:element name="externalPolicyRulesTemplate"
                      type="externalPolicyRulesTemplateType"
                      minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
```

```
            <xs:attribute name="templateID" type="xs:string"
                                    use="required"/>
            <xs:attribute name="templateVersion" type="xs:string"
                                use="required"/>
</xs:complexType>

<xs:complexType name="externalPolicyRulesTemplateType" abstract="true">
            <xs:sequence>
            </xs:sequence>
</xs:complexType>
```

The externalPolicyRulesTemplateType contains a template for any type of policy rules sent to the PEEM engine in real time and an implementation based on PEEM using this optional feature can define the structure carrying the policy rule data by extending this.

Declare the container element (policyInputData) to contain the element (inputTemplate), which is of the abstract types as defined above:

```
<xs:element name="policyInputData">
     <xs:complexType>
          <xs:sequence>
              <xs:element name="policyInputTemplate"
                   type="inputTemplateType"
                   minOccurs="1" maxOccurs="unbounded"/>
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

In an instance document, the content of <inputTemplate> can only be of a concrete type which derives from inputTemplateType, such as OtherEnablerInputType or PEEMInputTemplateType. The same extension mechanism is used for the <externalPolicyRulesTemplate>.

With this method instance documents will always contain the same elements (policyInputTemplate). However, these elements can contain variable content.

The following example schema for a new template illustrates how extensions are built using XML schema extension mechanism for types. Note that “new-input-template-namespace-id” needs to be replaced by the namespace identifier of the new template. Further note that the type “XYZ” serves as a placeholder for a data type to be defined by the application.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:pem1-i="urn:oma:xml:peem:pem1-input-template:1.0"
    xmlns="new-input-template-namespace-id"
    targetNamespace="new-input-template-namespace-id">

    <xs:import namespace="urn:oma:xml:peem:pem1-input-template:1.0"
```

```
schemaLocation="http://www.openmobilealliance.org/tech/profiles/PEM_1_Generi
cInputTemplateData-v1_0.xsd"/>

    <xs:complexType name="OtherEnablerInputType">
          <xs:complexContent>
               <xs:extension base="pem1-i:inputTemplateType">
                    <xs:sequence>
                        <xs:element name="xyz" type="XYZ"/>
                    </xs:sequence>
               </xs:extension>
          </xs:complexContent>
     </xs:complexType>
</xs:schema>
```

XML Document will then look like this:

```
<?xml version="1.0"?>
<pem1-i:policyInputData
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:pem1-i="urn:oma:xml:peem:pem1-input-template:1.0"
               xmlns:new-i="new-input-template-namespace-id"
               xsi:schemaLocation="new-input-template-namespace-id
                   location-of-new-input-template-schema.xsd">

     <policyInputTemplate xsi:type="pem1-i:PEEMInputTemplateType"
               templateID="PEEMTemplateID_1"
               templateVersion="V1.0.0">
     </policyInputTemplate>

     <policyInputTemplate xsi:type="new-i:OtherEnablerInputType"
               templateID = "otherEnablerInputTemplateID_2"
               templateVersion = "V1.0.0">
          <xyz>XYZ Type data</xyz>
     </policyInputTemplate>

     <!-- ... further templates if needed ...-->

</pem1-i:policyInputData>
```

C.1.2 Implementation Output Template

Define the abstract base type (outputTemplateType):

```
<xs:complexType name="outputTemplateType" abstract="true">
     <xs:sequence>
           <xs:element name="StatusCode" type="xs:unsignedShort"
                 minOccurs="1" maxOccurs="1"/>
           <xs:element name="StatusText" type="xs:string"
```

```
                    minOccurs="0" maxOccurs="1"/>
      </xs:sequence>

      <xs:attribute name="templateID" type="xs:string"
                              use="required"/>
      <xs:attribute name="templateVersion" type="xs:string"
                              use="required"/>
</xs:complexType>
```

Declare the container element (policyOutputData) to contain an element (outputTemplate), which is of the abstract type defined above:

```
<xs:element name="policyOutputData">
      <xs:complexType>
          <xs:sequence>
              <xs:element name="policyOutputTemplate"
                    type="outputTemplateType"
                    minOccurs="1"
                    maxOccurs="unbounded"/>
          </xs:sequence>
      </xs:complexType>
</xs:element>
```

In an instance document, the content of <policyOutputTemplate> can only be of a concrete type which derives from outputTemplateType, such as OtherEnablerOutputType or PEEMOutputTemplateType.

With this method instance documents will always contain the same element (outputTemplate). However, these elements can contain variable content.

The following example schema for a new template illustrates how extensions are built using XML schema extension mechanism for types. Note that “new-output-template-namespace-id” needs to be replaced by the namespace identifier of the new template. Further note that the type “XYZ” serves as a placeholder for a data type to be defined by the application.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:pem1-o="urn:oma:xml:peem:pem1-output-template:1.0"
    xmlns="new-output-template-namespace-id"
    targetNamespace="new-output-template-namespace-id">

    <xs:import namespace="urn:oma:xml:peem:pem1-output-template:1.0"
            schemaLocation="http://www.openmobilealliance.org/tech/profiles/
PEM_1_GenericOutputTemplateData-v1_0.xsd"/>

    <xs:complexType name="OtherEnablerOutputType">
          <xs:complexContent>
              <xs:extension base="pem1-o:outputTemplateType" >
                    <xs:sequence>
                        <xs:element name="xyz" type="XYZ"/>
                    </xs:sequence>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

XML Document will then look like this:

```
<?xml version="1.0"?>
<pem1-o:policyOutputData
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:pem1-o="urn:oma:xml:peem:pem1-output-template:1.0"
     xmlns:new-o="new-output-template-namespace-id"
     xsi:schemaLocation="new-output-template-namespace-id
                  location-of-new-output-template-schema.xsd">

     <policyOutputTemplate xsi:type="pem1-o:PEEMOutputTemplateType"
               templateID="PEEMTemplateID_1"
               templateVersion="V1.0.0">
          <StatusCode>2101</StatusCode >
          <StatusText>ALLOW</StatusText >
     </policyOutputTemplate>

     <policyOutputTemplate xsi:type="new-o:OtherEnablerOutputType"
                templateID="otherEnablerOutputTemplateID_2"
                templateVersion="V1.0.0">
          <StatusCode>2101</StatusCode >
          <StatusText>ALLOW</StatusText >
          <xyz>XYZ Type  data</xyz>
     </policyOutputTemplate>

     <!-- ... further templates if needed ...-->

</pem1-o:policyOutputData>
```

Appendix D. Common PEM-1 status definitions for use with Output Status Template (Normative)

This section defines the status that can be returned by a PEEM implementation, in response to a PEM-1 request. PEM-1 Status as defined in this section SHALL be supported as a minimum by a PEEM implementation, when it uses Output Status Templates. Additional status values MAY be specified according to D.6 below when needed.

A PEM-1 status is composed of a mandatory status code (integer) and an optional status text (string).

When the PEM-1 request involves a policy that uses an Output Status Template in the response, the status code SHALL be passed in the statusCode parameter of the Output Status Template, and an optional status text MAY be passed in the statusText parameter of the Output Status Template. In addition to that, status codes and status text defined here SHALL be exposed through different bindings using methods specific to those bindings.

When the PEM-1 request uses an Output BLOB without defining an Output Status Template within the BLOB, the status codes defined below are informative.

PEM-1 status codes can be in one of 5 possible classes, with pre-defined ranges as defined below:

- 1xxx (Informational). Status codes that fall within this category are used to inform the requestor that a request could not be satisfied, and additional action is required on its part before access is granted. No such status codes have been yet defined.

- 2xxx (Normal Evaluation). Status codes that fall within the Normal Evaluation category are used to inform a requestor that a request has been successfully completed with a result of either one of the types ALLOW, DENY or SUCCESS.

- 3xxx (PEEM Protocol Errors). Status codes that fall within this category are used to inform the requestor that a an application protocol issue has been identified by PEEM, in the case it was not identified when handling errors at the specific protocol binding layer.

- 4xxx (Transient Failures). Errors that fall within the Transient Failures category are used to inform a requestor that the request could not be satisfied at the time it was received, but MAY be able to be satisfied in a future attempt.

- 5xxx (Permanent Failures). Errors that fall within the Permanent Failures category are used to inform the requestor that the request failed, and should not be attempted again prior to addressing the issues identified.

A non-recognized class (one whose first digit is not in the range 1-5) MUST be handled as a permanent failure.

D.1 Informational Status (1000-1999)

Status Code	(Proposed) Status Text	Description
1000-1100 are reserved.		
1101-1200 are reserved for the use of future PEEM releases.		

D.2 Normal Evaluation Status (2000-2999)

Status Code	(Proposed) Status Text	Description
2000-2100 are reserved.		
2101	ALLOW decision	Policy processing resulted in an ALLOW decision
2102	ALLOW decision with	Policy processing resulted into

Status Code	(Proposed) Status Text	Description
	additional results	an ALLOW decision; additional results in the Output Status Template
2103-2110 reserved for “ALLOW” type codes in future PEEM releases		
2401	DENY decision	Policy processing resulted in a DENY decision
2402	DENY decision with additional results	Policy processing resulted into a DENY decision; additional results in the Output Status Template
2403-2410 reserved for “DENY” type codes in future PEEM releases		
2701	SUCCESS	Policy processing was successful, but it did not require rendering an ALLOW or DENY decision or it will be done later.
2702	SUCCESS with additional results	Policy processing was successful, but it did not require rendering an ALLOW or DENY decision or it will be done later; additional results in the Output Status Template
2703-2710 reserved for “SUCCESS” type codes in future PEEM releases		

D.3 Protocol Errors Status (3100-3999)

Status Code	(Proposed) Status Text	Description
3000-3100 are reserved		
3101-3200 are reserved for future PEEM releases		

D.4 Transient Failures Status (4100-4999)

Status Code	(Proposed) Status Text	Description
4000-4100 are reserved		
4101	Out of disk space	PEEM implementation ran into an out-of-disk-space error
4102	Out-of-memory error	PEEM implementation ran into an out-of-working-memory error

Status Code	(Proposed) Status Text	Description
4103	Server busy	PEEM implementation is temporarily overloaded
4104	Internal server error %	Internal server error % occurred.
4105	Operation aborted for reason %	PEEM processing was aborted for reason %
4106	Request time too skewed	Time interval associated with the request expired before completion of processing.
4107-4200 are reserved for future PEEM releases		

D.5 Permanent Failures Status (5100-5999)

Status Code	(Proposed) Status Text	Description
5000-5100 are reserved		
5101	Unsupported Input Template	An Input Template that is not supported for the policy was detected in the Policy-Data AVP
5102	Unsupported Input Template Version	An Input Template that is not supported for the policy was detected in the Policy-Data AVP
5103	Malformed Policy Data	Failure in parsing the XML document carrying the policy data.
5104	Invalid input parameter value in Input Template	A valid Input Parameter in the Input Template has an invalid value type.
5105	Missing Mandatory parameter(s) in Input Template %	Mandatory parameters (s) in the Input Template was/were missing. Possible indication of missing parameters via %
5106	Invalid input parameter value.	A valid Input Parameter in the Input Template has a value out of the expected range.
5107	Conflicting parameters in Input Template	Mutually exclusive parameters were detected in the Input Template
5108	Unsupported parameter in Input Template	An unsupported parameter was detected in the Input Template
5109	Optional External Policy feature not supported	PEEM implementation does not support this optional feature.

Status Code	(Proposed) Status Text	Description
5110	External Policy Reference not found	The External Policy Reference URI passed in the Input Template could not be found.
5111	External Policy invalid schema	The External Policy provided has a schema that does not conform to the PEL schema supported by the PEEM implementation
5112	Internal Policy Reference not found	The Internal Policy Reference URI passed in the Input Template did not match any known internal policy.
5113	Unspecified error	An unspecified error was detected. Check for additional results in the Output Status Template.
5114	Use of Call Back mode is not allowed	The use of call back is not allowed for this application.
5115	The Call Back mode is not supported	The use of call back is not supported by the Server.
5116	The Call Back provided is not OK (%)	The provided call back address is not valid.
5117-5200 are reserved for future PEEM releases		

D.6 Extensions

Additional status codes MAY be defined in the 5 defined classes, using codes that follow the ones that have been defined in D.1 through D.5, in the following ranges:

Additional Informational status codes range: 1201-1999

Additional Success status codes range as follows:

- ALLOW type codes in the range: 2111-2400
- DENY type codes in the range: 2411-2700
- SUCCESS type codes in the range: 2711-2999

Additional Protocol Errors status codes range: 3201-3999

Additional Transient Failures status codes range: 4201-4999

Additional Permanent Failures status codes rage : 5201-5999

Any OMA Working Group, service provider or vendor MAY add additional errors in the classes described above. The unused specified ranges in the sections D.2 through D.5 are reserved for future releases of the PEEM specifications.

Appendix E. A guideline for defining templateID and templateVersion (Informative)

Templates (Standard or Custom) need to be uniquely identified, using a combination of templateID and templateVersions (see normative section 5.2). However, other than ensuring that the combination of the two strings used is unique, there is no need to specify and therefore limit how templateID and templateVersion are defined. Instead, this Appendix provides a guideline for the definition of PEM-1 templateID and templateVersion.

Since new Standard PEM-1 Templates may be defined by different enablers, and Custom PEM-1 Templates may be defined by Service Providers, the proposal is to use a combination of OMA unique identifiers when defining the strings for templateID and templateVersion.

The recommendation is that the standard templateID string be composed by concatenating (separated by '_'):

- OMA
- A unique Enabler identifier
- A unique sequentially increasing number for the template (assuming multiple templates are defined for the same WI and same enabler)
- An optional name

An example using this recommendation would be: 'OMA_PEEM_1'.

For custom PEM-1 templates, they should not be prefixed with 'OMA'.

The recommendation is that the templateVersion string is represented by a using the current OMA convention for versioning (i.e. 3 digits, pre-pended by the letter V). An example using this recommendation would be 'V1.0.0'.

Appendix F. Communicating PEM-1 details to the requestor (Informative)

F.1 Use cases

Interpretation of the BLOB input data structure and generation of output data is always driven by the policy that is processed. If a PEM-1 Template is used within a BLOB, as described in section 5.1.2, the data structure is expected to follow the PEM-1 Template and the policy is expected to be designed to interpret the data structure accordingly.

To facilitate management and interaction by a requestor, PEM-1 Templates may be used to determine the BLOB internal format. In such case all or a subset of policies are expected to follow a specific PEM-1 Template. PEM-1 Templates can be Standard (included with the PEEM specification) or Custom (e.g. vendor specific, defined by the Service Provider which deploys PEEM). Inputs and outputs to be provided as part of the PEM-1 Template are determined by an established convention put in place to meet the needs of the policy, while considering the constraints of the requestor.

In order to use PEEM in callable mode, the PEEM requestor is expected to be aware of the input it needs to provide, and the output behaviour. How this is achieved is out of scope of PEEM specifications, but it is assumed to be communicated in a separate communication channel.

When PEM-1 Templates are not used, until the policies' expected input and output are defined and made available to a requestor, he may not know the data structure of the input to generate and output to expect.

When using PEM-1 Templates the PEEM requestor may know the data structure of the input to be generated and of the output that is expected as a response, before the policies are actually produced.

With certain PEM-1 Templates, until the policies' logic and variables are defined and made available the PEEM requestor may not always know the complete data structure of the input to generate, or the complete data structure of the output to expect as a response. This may be the case if a PEM-1 Template does not fully specify each and every input/output PEM-1 Parameter to be exchanged (e.g. the precise number of input/output PEM-1parameters for each category of PEM-1 Parameters expected). In this latter case, the use cases will show different ways of handling such a case:

- The SP and the implementations can have their own proprietary understandings/conventions of what data to put into the PEM-1 Templates and how to interpret that data
- The policy provides definition to input/output PEM-1 Parameters needed, but left undefined by the PEM-1 Template

The I/O data structure has to be communicated to the PEEM requestor. See examples in F.1.1 and F.1.2 on how to achieve that.

The following describe use cases or approaches that can be used to address these challenges.

F.1.1 Template selection

A service provider can limit its policies to follow a (or a few) Standard or Custom PEM-1 Template(s). The details (the complete PEM-1 Templates) are communicated to the requestor:

- At the time the policy's design is complete and therefore all input and output PEM-1 Parameters are determined
- At authoring of the applications or at subscription to the an exposed service (via a separate manual or automate mechanism (e.g. discovery))
- At execution of the application (via a separate manual or automate mechanism (e.g. discovery))

F.1.2 BLOB

A service provider can decide not to follow any PEM-1 Template (each policy may expect different input and generate different output). The details for each case are still to be communicated to the requestor:

- At the time the policy's design is complete and therefore all input and output PEM-1 Parameters are defined
- At authoring of the applications or at subscription to the an exposed service (via a separate manual or automate mechanism (e.g. discovery))
- At execution of the application (via a separate manual or automate mechanism (e.g. discovery))

F.2 Best Practices / Guidelines

The input is interpreted by the policy and output details are determined by the policy.

From the requestor point of view, the input and output details are defined by pre-determined PEM-1 Parameters (in the case a PEM-1 Template is being used).

In some cases the input and output details are defined by a combination of pre-determined PEM-1 Template including defined PEM-1 Parameters in combination with additional parameters needed as determined by the logic of the policy. This combination also needs to take into account the constraints of the requestor (some parameters desired by the policy may not be always be provided by the requestor).

Figure 1 below illustrates through an informative flow the use of the PEM-1 interface when passing a single string BLOB containing a PEM-1 Template. The details of the evaluation process in the PEEM enabler implementation and the enforcement process of the decision in the Policy Evaluation Requestor have been left out intentionally.

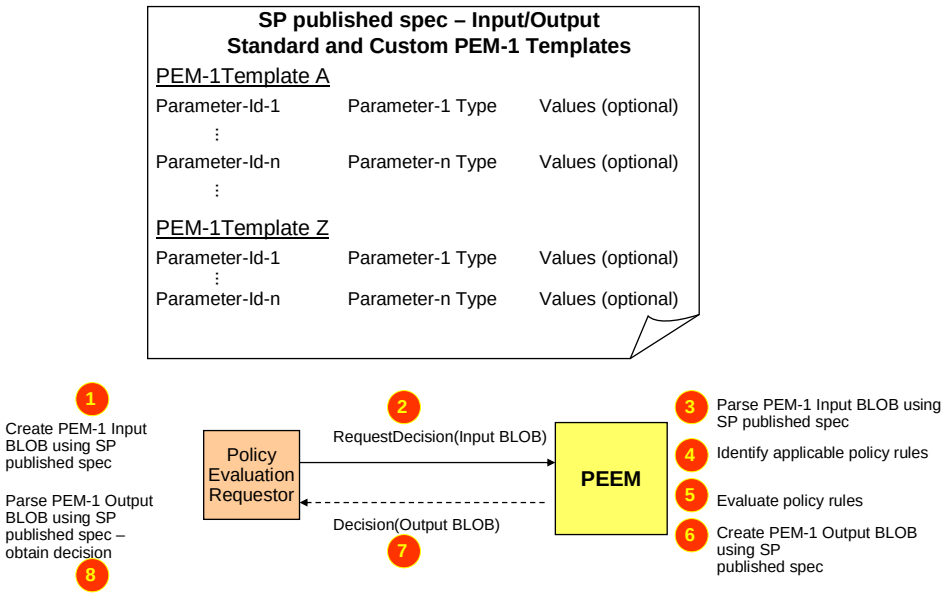


Figure 4: Handling input/output-policy-data as encapsulated PEM-1 templates in a BLOB

A Policy Evaluation Requestor has access to SP published/supported PEM-1 Templates specification (the specification follows the PEM-1 TS, but the form it is represented and accessed in the PEEM enabler implementation is out-of-scope for the PEM-1 TS). All the steps that make up flow 1 can be performed at runtime or ahead of runtime. The Policy Evaluation Requestor selects a PEM-1 Template applicable to its application and uses the published specification to obtain the PEM-1 Parameters that it needs to pass, their types and optionally, allowable values. It then encodes each of the attributes by concatenating them as a list of identifiable parameters, each followed by an actual associated parameter value (flow 1) to

form the PEM-1 Input BLOB Parameter. It then uses the protocol of choice, out of those supported by the PEM-1 TS specification, to forward the request for evaluation, including the single PEM-1 Input BLOB parameter (flow 2). The PEEM enabler implementation receives the request using the binding to the supported protocol. It extracts the PEM-1 Input BLOB Parameter and parses it with the help of the SP published/supported PEM-1 Templates specification (flow 3). That specification allows the PEEM implementation to know how to interpret each attribute in the PEM-1 Input BLOB Parameter..It identifies the PEM-1 Template used by the Policy Evaluation Requestor, to determine what PEM-1 Parameters may be expected. It may identify a PEM-1 Parameter that references an internal or external policy to be used (see section 5.2.2) in order to identify the applicable policy rules (flow 4). PEEM enabler implementation then processes the evaluation request which may result in a response (flow 5). The response is then encoded in a PEM-1 Output BLOB Parameter, again with help from the information available from a SP published/supported PEM-1 Template specification (flow 6). The response is sent using the selected protocol, to the Policy Evaluation Requestor (flow 7). The Policy Evaluation Requestor parses the PEM-1 Output BLOB Parameter, using the SP published/supported PEM-1 Templates specification and obtains the decision issued by PEEM (flow 8).

This mechanism allows the entire PEM-1 Template to be passed as a single interface parameter (a PEM-1 Input BLOB or PEM-1 Output BLOB) by any protocol chosen to support the PEM-1 requests and responses. Both input PEM-1 Template and output PEM-1 Template are handled in a similar way, although the content of the templates may be quite different, according to the PEM-1 Template definition. This allows complete decoupling of the PEM-1 interface specification from the particular PEM-1 Templates that it needs to transport, and from the particular PEM-1 Parameters inside the templates. It also supports the principle of neutrality to technology, since a binary string parameter (a PEM-1 BLOB Parameter) is the only mandatory parameter that needs to be transported over any binding, and any considered binding for the PEM-1 TS supports the passing of a string data type. Furthermore, this also significantly reduces the complexity of mapping the interface to different bindings, and provides a true scalable way to deal with adding new PEM-1 Templates and parameters. Finally, supported by a specified XML schema, this is the most convenient way to transport parameters, and alleviates the need on policy evaluation requestors and on the PEEM enabler implementations of stopping/re-compiling/re-starting a deployed system, since the PEM-1 interface does not have to change; the only adaptation needed for a PEEM implementation and/or the policy evaluation requestors is to be able to interpret and handle the content of a PEM-1 Template, using standard XML tools. That ensures stability of an implementation for the one part of the implementation that handles the communication protocol, and moves the burden of adaptability to the part that needs to deal with the understanding of the PEM-1 parameters, which is unavoidable anyway, since new policies, with new parameters, need to be continuously supported. The encoding scheme of PEM-1 Parameters into PEM-1 BLOB Parameters is described in the following section. The mapping to and use of the PEM-1 Input BLOB Parameter and PEM-1 Output BLOB Parameter for specific bindings are described in section 5.4.

Appendix G. I/O parameters (Informative)

This section is provided as guidance to how to construct I/O parameters for Standard PEM-1 and/or Custom PEM-1 Templates. Input/output parameters listed are non-exhaustive, and may be used in multiple Templates, if and as needed, to complete the Standard PEM-1 Templates documented in the normative sections of the document. For convenience, I/O parameters have been grouped here by the nature of the information they convey (e.g. parameters relative to template identification, originator identity, etc). The datatypes of I/O parameters have not been indicated here, since these parameters are only recommended, hence the parameters themselves, and their types may be added at will by the Service Provider and Vendors. PEM-1 templates may contain different combinations of parameters specified in this document, as well as any other parameters needed by the policy.

G.1 Origin-Identification

This section provides guidance on how to pass information about the origin and identity related to the original request for access to a resource (the resource being the one that invokes the help of PEEM). This may include information about a possible principal (e.g. end-user), the device the principal is using, and the application used by the principal to make the request. The following parameters may be used in the templates when conveying this type of information:

- OriginatorID – a parameter that identifies a principal that issued a request, or on behalf of whom a request was issued (name, pseudonym, other)
- OriginatorDomain – a parameter that identifies the originating principal's domain (realm)
- OriginatorDeviceID – a parameter that identifies the originating principal's device
- OriginatingApplicationID – a parameter that identifies the application via which the request for accessing a resource was made (ApplicationIDs would be assigned by the Service Provider and must be unique within the scope of that Service Provider)
- OriginatingApplicationDomain – a parameter that identifies the domain from which the application made the request

G.2 Target-Identification

This section provides guidance on how to pass information about the destination and identity related to the original request for access to a resource (the resource being the one that invokes the help of PEEM). Thi may include information about a possible principal (e.g. end-user), the device the principal is using, and the application used by the principal to make the request. The following parameters may be used in the templates when conveying this type of information:

- TargetID – a parameter that identifies a principal that is the target of a request (name, pseudonym, other)
- TargetDomain – a parameter that identifies the target principal's domain (realm)
- TargetDeviceID – a parameter that identifies the target principal's device
- TargetApplicationID – a parameter that identifies the application via which the target principal may be reached (ApplicationIDs would be assigned by the Service Provider and must be unique within the scope of that Service Provider)

TargetApplicationDomain – a parameter that identifies the domain in which the target application operates.

G.3 Resource-Identification

This section provides guidance on how to pass information about the resource that needs policy enforcement. This may include information useful in identifying the resource that issues a request to PEEM, the operation that was requested from this resource by some other application, the type of service that is involved in that original request. The following parameters may be used in the templates when conveying this type of information:

- ResourceID – a parameter that identifies the resource that is accessed by the originating principal (or an application representing that principal). This is the resource that issues the request towards PEEM (ResourceIDs would be assigned by the Service Provider and must be unique within the scope of that Service Provider)

- ResourceDomain – a parameter that identifies the domain in which that resource resides
- RequestedOperation – a parameter that identifies the request that was made against this resource
- RequestType – a parameter that categorizes the type of request that was made against this resource (e.g. end-user to end user, end-user to group, etc).

G.4 Charging-Identification

This section provides guidance on how to pass information about the entity that would be potentially charged in conjunction with handling a policy evaluation request. The following parameters may be used in the templates when conveying this type of information:

- ChargedPrincipalID – a parameter that identifies the principal that should be charged in conjunction with this request
- ChargedPrincipalDomain – a parameter that indicates the domain to which the charged principal belongs

G.5 Environment-Identification

This section provides guidance on how to pass state information about the environment in which the request to the resource has been made. The following parameters may be used in the templates when conveying this type of information:

- TimeofDay – a parameter that defines the time-of-day the original request was made
- OriginatorSphere – a parameter that defines the originating principal's environment (home, work, other)