PEEM Policy Expression Language Technical Specification

Approved Version 1.0 – 24 Jul 2012

Open Mobile Alliance

OMA-TS-PEEM_PEL-V1_0-20120724-A

Contents

1. Scope

This document provides the Policy Expression Language (PEL) specification, one of several specifications of the Policy Evaluation,Enforcement and Management (PEEM) enabler. The PEL specification defines the language in which policies can be expressed. The PEL specification includes the definition of language constructs, and defines multiple language options, for the convenience of resolving particular issues. The PEL specification is loosely coupled to other PEEM specifications and therefore can evolve relatively independent of them (see Introduction section for details). While PEL supports the expression of any policy, specific policies expressed in PEL are out-of-scope for the PEL specification.

2. References

2.1 Normative References

[BPEL] "Business Process Execution Language", OASIS,
URL: Web Services Business Process Execution Language Version 2.0
http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf

[IM_TS] "Instant Messaging using SIMPLE", Version 1.0, Open Mobile Alliance™, OMA-TS-SIMPLE_IM-V1_0, URL: http://www.openmobilealliance.org/

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

[MSRP] IETF draft-ietf-simple-message-sessions-19 "The Message Session Relay Protocol", B. Campbell, R. Mahy, C. Jennings, February 2007, URL: http://www.ietf.org/internet-drafts/draft-ietf-simple-message-sessions-19.txt

Note: Work in progress

[PEEM AD] "Policy Evaluation, Enforcement and Management Architecture", Open Mobile Alliance™, Version 1.0, OMA-AD_Policy_Evaluation_Enforcement_Management-V1_0,
URL: http://www.openmobilealliance.org/

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

[PEEM_Callable_Policy_Interface] "Policy Evaluation, Enforcement and Management Callable Interface (PEM-1) Technical Specification", Open Mobile Alliance™, OMA-TS-PEEM_PEM1-V1_0, URL: http://www.openmobilealliance.org/

[PoC_CP] "Push to talk Over Cellular (PoC) - Control Plane Specification", Version 2.0, Open Mobile Alliance™, OMA-TS-PoC-ControlPlane-V2_0, URL: http://www.openmobilealliance.org/ "

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

[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 3428] IETF RFC 3428 "Session Initiation Protocol (SIP) Extension for Instant Messaging", B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, December 2002, URL: http://www.ietf.org/rfc/rfc3428.txt

[RFC 3840] IETF RFC 3840 "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", J. Rosenberg, H. Schulzrinne, P. Kyzivat, August 2004, URL: http://www.ietf.org/rfc/rfc3840.txt

[RFC 4745] "Common Policy: A Document Format for Expressing Privacy Preferences, H.Schulzrinne et al, IETF RFC 4745, February 2007, URL: http://www.rfc-editor.org/rfc/rfc4745.txt

[RFC 4825] IETF RFC 4825 "The Extensible Markup Language (XML) Configuration Access protocol (XCAP)", J. Rosenberg, May 2007, URL: http://www.ietf.org/rfc/rfc4825.txt

[XSD_BPEL] OASIS, WS-BPEL 2.0 XSD, http://docs.oasis-open.org/wsbpel/2.0/CS01/process/executable/ws-bpel_executable.xsd

[XSD_commPol] "XML Schema Definition: "XDM – Common Policy", Version 1.0, Open Mobile Alliance™, OMA-SUP-XSD_xdm_commonPolicy-V1_0, URL: http://www.openmobilealliance.org/

[XSD_ext] "XML Schema Definition: XDM Extensions", Version 1.0, Open Mobile Alliance™, OMA-SUP-XSD_xdm_extensions-V1_0, URL: http://www.openmobilealliance.org/

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

[XDMSPEC] "XML Document Management (XDM) Specification", Open Mobile Alliance™, OMA-TS-XDM_Core-V2_0, Candidate Version 2.0, URL: http://www.openmobilealliance.org/

[XPATH1.0] W3C Recommendation, "XML Path Language (XPath) Version 1.0", J. Clark, S. DeRose, November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116

2.2 Informative References

[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

[WP-LOCRULES] "Geolocation Policy: A Document Format for Expressing Privacy Preferences for Location Information", H.Schulzrinne et al, IETF draft, May 22, 2007, URL: http://tools.ietf.org/id/draft-ietf-geopriv-policy-12.txt

[WP-PRESRULES] "Presence Authorization Rules", J.Rosenberg, IETF draft, February 27, 2007, URL: http://www.ietf.org/internet-drafts/draft-ietf-simple-presence-rules-09.txt

3. Terminology and Conventions

3.1 Conventions

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

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

3.2 Definitions

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

3.3 Abbreviations

AVP	Attribute Value Pair
BPEL	Business Process Execution Language
OMA	Open Mobile Alliance
PEEM	Policy Evaluation, Enforcement and Management
PEL	(PEEM) Policy Expression Language
PEM-1	PEEM Callable interface
PEM-2	PEEM Management interface
PoC	Push to talk over Cellular
URI	Uniform Resource Identifier
WS	Web Service(s)
WS-BPEL	Web Services Business Process Execution Language (synonym to BPEL)
WSDL	Web Service Definition Language
XDM	XML Document Management
XML	eXtensible Markup Language

4. Introduction

The PEL specification explains and defines language options, constructs, variables, constants and operators needed to express conditions and actions, and a detailed syntax to express them. The PEL specification is built to support OMA enablers' policy needs, as well as needs of other resources. Recognizing the fact that different requirements apply, the PEL specification supports at least two separate options of expressing conditions and actions: a ruleset-based option, and a workflow-based option. In the ruleset-based option, each rule is evaluated as separate entity, and the combination of the results of the processing of all the rules in the ruleset determines the policy outcome (notice that a precedence mechanism may be needed). In the workflow-based option, the entire policy is processed as a whole, following a flowchart approach, where at each node in the graph, a rule is being processed.

The PEL specification can evolve independent of PEM-1 and PEM-2.

Finally, resources that make use of PEEM may or may not be aware of the PEL policies or the PEL options deployed, but they do not have to be aware of them.

5. Policy Expression Language

Policy expression languages are designed to cover specific usage scenarios. For example a ruleset language may be a better approach for preferences and permissions related policies, while a language for business process execution may be a better approach for orchestration and delegation to other resources. Since PEEM is a generic enabler that could be deployed in various scenarios, it is beneficial to specify the best Policy Expression Language that is appropriate to describe policies for the particular deployment situation. Two Policy Expression Language options (a PEL ruleset language option, and a PEL for business process option) comprise the PEL specification.

PEEM enabler implementations SHALL offer at least one of the following Policy Expression Languages:

- A ruleset language
- A business process execution language

Other languages, or correlations between options, are not precluded, but are not described in this specification.

5.1 PEL Ruleset Framework

The PEL ruleset language option is based on the ruleset framework described in IETF RFC 4745 [RFC 4745]. The use of this framework had been suggested in PEEM Architecture Document [PEEM AD], while at the time that framework was being worked as an IETF draft. The policy framework described in IETF RFC 4745 [RFC 4745] is already used and extended by some OMA enablers (see section 5.3.4), and its re-use by PEL will facilitate adoption in OMA.

The PEL ruleset language option in this release SHALL adhere to the provisions in RFC 4745 [RFC 4745] and to the XML Common Policy schema included with the RFC 4745. Policies written using the ruleset language option SHALL conform to the XML schema defined in [RFC 4745] and extended in OMA with [XSD_commPol] and [XSD_ext]. The use of the RFC 4745 <transformation> element is optional for the PEEM Policy Expression Language, since the PEEM model of policy evaluation and enforcement does not require the use of transformation, since those can be captured in the actions themselves.

In addition to adherence to RFC 4745, PEL ruleset language option SHALL include extensions added to RFC 4745 in [XDMSPEC], for the purpose of supporting policy-related functionality in other OMA enablers (see 5.1.4 for details).

The following provisions in RFC 4745 are however considered restrictive, and the restrictions they impose SHOULD be addressed by PEL future extensions (not included in PEEM release 1.0):

1. The optimization provided by RFC 4745 [RFC 4745] to only allow “permit” style rules SHOULD be revisited in the context of specific policies for enablers that rely on PEEM (i.e. future extensions for “deny” style rules MAY be considered when needed). When added, the semantics for rules including both “permit” and “deny” style rules need to be fully understood to avoid conflicts.
2. The optimization provided by RFC 4745 [RFC 4745] to specify a particular algorithm of combining “permissions” (actions and transformations) SHOULD be revisited in the context of specific policies for enablers that rely on PEEM (i.e. future extensions MAY be provided to allow different algorithms to evaluate rulesets, or alternatively this will be left to differentiate implementations).

This section provides a summary of RFC 4745 [RFC 4745] framework. The details of each framework construct are normatively described in [RFC 4745]. This current version may not meet all the PEEM needs for a PEL, but it provides an extensible framework which allows it to progress in time towards the full needs of a generic PEL. The extensibility model is on a need-basis, as determined by specific applications domain, allowing this language option to progress in sync step with the needs of OMA enabler and other resources, as expressed by their specific requirements. A comparison between PEEM PEL needs and RFC 4745 [RFC 4745] is provided in Appendix C. Extensions determined necessary in other OMA enablers or other resources may be provided either as part of such future enablers extensions to PEL ruleset option, or in a future PEEM PEL phase (currently not planned). If such extensions are needed before they materialize into OMA extensions to PEL, it is expected that vendors may provide proprietary extensions to PEL ruleset language to fulfill the need.

5.1.1 Overview

RFC 4745 [RFC 4745] is combining two authorization systems (for presence and location) into a more generic framework, with mechanisms for extensibility. This general framework is intended to be accompanied and enhanced by other domain-specific policy documents, including presence [WP-PRESRULES] and [WP-LOCRULES] (these are "work-in-progress" examples of how to extend RFC 4745, and from PEL TS perspective are only informative at this point in time).

The current applicability of RFC 4745 [RFC 4745] is not limited to policies controlling access to presence and location information data, but can be extended to other applications domains.

The mode of operation supported by RFC 4745 [RFC 4745] can be described as very similar with the PEEM component [PEEM AD] behaviour. A Policy Authorization Server receives a query regarding data items for a particular requestor, via the using protocol (i.e. the policy invocation protocol, equivalent to PEM-1 interface in PEEM). The using protocol provides parameters (e.g. identity of the requestor, etc). The input information, together with additional data accessible by the Policy Authorization Server is used for searching through a ruleset, defined using the RFC 4745 [RFC 4745] framework. All matching rules are combined according to a specified permission combining algorithm. The combined rules are applied leading to results that are being returned via the using protocol to the requestor.

There are three different modes of operation supported, passive request-response, active request-response and event notification. The passive request-response mode in RFC 4745 [RFC 4745] matches the PEEM callable pattern.

The framework in its current version provides construct based on a simplifying pre-condition to its design, namely that each rule must be representable as a row in a relational database, to allow for efficient policy implementation by utilizing standard database optimization techniques. This pre-condition explains decisions made in the design of the Common Policy framework constructs.

Another design consideration is that the current version only provides permissions rather than denying them (i.e. removing a rule can never increase permissions). That design consideration was also in order to optimize implementation, by removing the concern about how to deal with ordering of the rules, and potential conflict between "deny" rules and "permit" rules (if both were to be allowed). Hence, only "permit" related actions are currently supported, and rules ordering is no longer important. By default, rules will be evaluated in the order they appear in the policy. At the same time, processing all the rules is instead required.

The framework assumes permissions are additive, in the sense that if several rules match, then the overall permissions granted to the requestor are the union of the permissions of all the rules that match.

The following describes in more detail how rules are combined (for complete details see [RFC 4745]):

- Each type of permission is combined across all matching rules.
- Each type of action or transformation is combined separately and independently.
- The rules are combined based on the type of permission.
    - For boolean permissions, the resulting permission is TRUE if and only if at least one permission in the matching rule set has a value of TRUE and the resulting permission is FALSE otherwise.
    - For integer, real-valued and date-time permissions, the resulting permission is the maximum value across the permission values in the matching set of rules.
    - For sets, the resulting permission is the union of values across the permissions in the matching rule set

The framework in RFC 4745 [RFC 47445] explicitly lists the following items as being out-of-scope for the current version:

- Access of external rulesets, databases, directories, or other network elements
- Support of regular expressions (i.e. conditions are matched on equality or "greater-than" style comparisons, not on regular expressions like those encountered in wild-card matches)

A ruleset (i.e. a policy) consist of zero or more rules. A rule consists of three parts: conditions, actions and transformations.

The conditions is a set of expressions, each of which evaluating to either TRUE or FALSE. Actions express the permitted output results, before applying transformations (e.g. DENY or PERMIT). The transformations apply when the action indicates permission, and they specify how information results are to be modified before being provided to the requestor.

Rules are encoded in XML, and RFC 4745 [RFC 4745] includes a schema defining the Common Policy Markup Language. The XML schema defines the exchange format between a requestor and the Policy Authorization Server, but it is clearly stated that there is no implication that such a schema will be used internally by either the requestor or the Policy Authorization Server. The rules are designed so that a Policy Authorization Server can translate them into a relational database table, with each rule represented by one row in the database. The database representation is also not mandatory; it is merely a well-understood example of internal representation, out of many other possible implementations. Extensions canot change the schema provided in RFC 4745 [RFC 4745]], and this schema is not expected to change in future versions, which explains why no versioning procedures exist. The RFC 4745 Common Policy framework assumes the extensions to come in the form of additional expressions of conditions, actions or transformations.

5.1.2 Constructs

The ruleset framework in RFC 4745 [RFC 4745] relies on ruleset, rule, conditions, actions and transformations as basic, extensible elements.

The ruleset is composed of several rules, and each rule includes conditions, actions and transformations. An instance of a request for policy (ruleset) evaluation provides several attributes, that need to be matched against variables used in conditions and actions in the rules of the ruleset.

The <condition> element is used to express conditions (a rule may include multiple <condition> elements). The framework only specifies a limited number of generic conditions, re-usable across different application domains (i.e. conditions for identity, sphere and validity attributes). Additional conditions will be specified elsewhere, with their own namespace. The <condition> element may have child elements, defined in this framework, or extended somewhere else. If a child element of the <condition> construct is in a namespace that is not known or not supported, then this child element evaluates to FALSE.

Conditions are matched on equality, or “greater-than” style comparison, according to the datatype associated with the element in the RFC 4745 [RFC 4745] schema. There are three <condition> child elements specified in this framework (identity, sphere and validity), and each may have additional child elements.

The <identity> element used in expressing conditions is considered TRUE if any of its child elements evaluate to TRUE. Child element supported are <one> (for matching a single authenticated identity) and <many> (for performing authorization decisions based on the domain portion of an authenticated identity. The <many> element supports the child element <except> which fulfills the need of excluding elements from the solution set.

The <sphere> element can be used to indicate an environment or context (e.g. “work”, “home”, “meeting”, “travel”, etc). A <sphere> condition matches only in the case of equality with the requestor’s environment or context passed in the request, or otherwise available to the ruleset.

The <validity> element is used to express the period of time interval in which the rule is valid. All the conditions in a rule MUST evaluate to TRUE in order for a rule to evaluate to TRUE; hence this element can be used to completely invalidate a rule, regardless of the outcome of the other conditions. This allows to provision rules whose validity is temporary, without a concern for an administrator for immediate maintenance (other than periodic cleanup of the expired rules).

In short, each type of condition determines its own semantics of evaliting to TRUE. A rule evaluates to TRUE when ALL conditions in its <condition> child elements evaluate to TRUE. All rules that have <condition> child elements that match (evaluate to TRUE) form the matching ruleset. The matching ruleset has an associated mechanism for combining the permissions, based on the <action> elements and <transformation> elements in the rule.

The <action> element is used to specify the policy output results, and the <transformation> element is used to apply changes (e.g. filters) to those results before forwarding them to the requestor. Together, the <action> and <transformation> elements are also referred to as “permissions”. The “permissions” are combined across all matching rules (i.e., evaluated to TRUE) in the ruleset. The combining rules depend on the datatype of the “permision”. For example, if the “permision” (action or transformation) datatype is boolean, then the resulting “permission” for the ruleset is TRUE if and only if at least one of the “permissions” in the matching ruleset is TRYE (in other words, it is an OR operation between all “permissions”). If the

"permission" is of datatype integer, or float for example, the resulting "permission" is is the maximum value from the set of individual "permission" values associated with each matching rule in the ruleset.

5.1.3 Language Extensibility and support of OMA specific conditions

RFC 4745 [RFC 4745] supports extensibility by allowing namespace-qualified extensions to be added in the form of <conditions>, <actions>, <transformations> and unrestricted number and variety of child elements of these elements. Schemas for the added elements and new namespaces need to be provided as part of providing such extensions. The RFC 4745 [RFC 4745] has an explicit model of adding extensions, and expects that such model will not infringe on the RFC 4745 [RFC 4745] schema, but expand it. The framework also has an explicit stated goal of having extensions provided in support of specific application domains, in order to justify the extensions, and not unnecessarily complicating the framework.

The framework makes no statement about the mandatory or optional nature of extensions.

The extensibility may be used to address any framework generic enhancements (e.g. additional constructs, or additional semantics of existing constructs). See Appendix C for a comparison between PEEM PEL needs and RFC 4745 [RFC 4745] features.

The extensibility may also be used to address any specific application domains needs (additional <condition>, <action>, <transformation> and their child elements) for:

- Security rules (e.g. Authentication, authorization, GPM, confidentiality (selective), integrity, …)
- Charging rules
- Logging rules
- Privacy rules
- Preference rules
- Permission rules
- Content screening rules
- Content categorization rules

5.1.4 Backward compatibility with other OMA enablers

Policy requirements have been addressed in several other OMA enablers. In most cases, the basis for the technical specifications addressing policy requirements is also IETF RFC 4745 [RFC 4745]. When using this policy framework, those enablers have introduced extensions in a manner consistent with the model described by IETF RFC 4745 [IETF 4745]. Extensions included new elements and new semantics associated with those elements, as well as over-riding assumptions made in IETF RFC 4745 [RFC 4745]. That work is completely consistent with the approach taken by PEL ruleset language option described in this section, and hence the PEL specification for ruleset language SHALL include the following extensions added in [XDMSPEC] to support OMA enablers:

- reference identities in external URI lists, which is an explicit non-goal of [RFC 4745];
- enable the user to define a default rule that applies in the absence of any other matching rule;
- allow rules to be matched based on hierarchical precedence assigned to the different types of allowed conditions, prior to combining permissions;
- constrain, for predictability in UE design and end user expectation, the conditions in a rule to no more than a single expression or set of expressions.

NOTE 1: Individual enablers may also define extensions to [RFC 4745] to meet application-specific needs. Such extensions must not change or cause to change the semantics of the common extensions defined in section 5.1.4.1 or the evaluation algorithm for combining permissions defined in section 5.1.4.4.

NOTE 2: An authorization policy using the extensions defined in this sub-clause must declare the "urn:ietf:params:xml:ns:common-policy" and "urn:oma:xml:xdm:ns:common-policy" namespace names in the XML schema.

5.1.4.1 Structure

The <conditions> element within a rule in an authorization policy:

1) MAY include the <identity> condition element as defined in [RFC 4745];
2) MAY include the <external-list> condition element;
3) MAY include the <anonymous-request> condition element;
4) MAY include the <other-identity> condition element;
5) MAY include the <media-list> condition element;
6) MAY include the <service-list> condition element.

The <conditions> element of a rule SHALL contain no more than one of <identity>, <external-list>, <anonymous-request> or <other-identity>, but it MAY contain other elements (i.e. a <media-list> element and a <service-list> element).

The <external-list> element MAY include the <entry> element. If present, the <entry> element SHALL include the "anc" attribute, whose value SHALL be percent-encoded as defined by [RFC 4825] section 6 before it is inserted into a document.

The <media-list> element SHALL include one of:

1) an <all-media-except> element or;
2) a list of one or more media elements selected from the list of possible media elements below.

List of possible media elements:

1) The <message-session> media element indicating session based messaging as defined in [MSRP];
2) The <pager-mode-message> media element indicating pager mode message requests as defined in [RFC 3428];
3) The <file-transfer> media element indicating file transfer as defined in [IM_TS];
4) The <audio> media element indicating a streaming media type as defined in [RFC 3840];
5) The <video> media element indicating a streaming media type as defined in [RFC 3840];
6) The <poc-speech> media element indicating a PoC speech media type as defined in [PoC_CP];
7) The <group-advertisement> media element indicating a group advertisement as defined in [XDM_Group] "*Extended Group Advertisements*";
8) Any elements from any other namespaces defining a media element.

The <all-media-except> element MAY include a list of one or more media elements selected from the list of possible media elements above.

The <audio>, <video> and <message-session> elements:

1) MAY include the <full-duplex> element indicating that media can be exchanged in both directions simultaneously;
2) MAY include the <half-duplex> element indicating that media can be exchanged in only one direction at a time.

The <service-list> element SHALL include one of:

1) a list of one or more <service> elements or;
2) an <all-services-except> element.

The <all-services-except> element MAY include a list of one or more <service> elements.

The <service> element:

1) MAY include an attribute “enabler” including a string defining an Enabler ;
2) MAY include attributes from any other namespace for the purpose of extensibility to have other service identification definitions than using the “enabler” attribute;
3) MAY include any other elements from any other namespaces for purpose of extensibility to have other service identification definitions than using attributes.

5.1.4.2 Data Semantics

If present in any rule, the <external-list> element SHALL match those identities that are part of a URI List.

If present in any rule, the <anonymous-request> element SHALL match those incoming requests that have been identified as anonymous.

NOTE 1: In certain cases, the <identity> condition can also match anonymous requests. For example, the <many/> child element of the <identity> condition matches any authenticated identity, either anonymous or not. However, any rules matching the <anonymous-request> condition would have precedence as described in section 5.1.4.3 “*Combining Permissions*”

NOTE 2: If the authorization policy document includes a rule having an <anonymous-request> condition element, an XDMC should not specify another rule containing an <identity> condition element with a <many/> child element and the same <actions> and/or <transformations> element(s) as the rule with the <anonymous-request> condition element.

If present in any rule, the <other-identity> element, which is empty, SHALL match all identities that are not referenced in any rule. It allows for specifying a default policy.

If present in any rule, the <media-list> element SHALL match incoming requests associated with particular media types. A <media-list> element with a list of media elements SHALL be used to specify allowed media types. A <media-list> element with an <all-media-expect> element SHALL be used to specify that all media types are allowed apart from those listed as child elements. The <media-list> condition SHALL be considered TRUE if any of its child media elements evaluate to TRUE, i.e., the results of the individual child elements are combined using a logical OR. The <media-list> condition SHALL also be considered TRUE if all of the child media elements to an <all-media-except> element evaluate to FALSE.

If neither a <full-duplex> nor <half-duplex> duplex specific sub element is included, it means that the access rule is applicable to both cases (i.e. half-duplex and full-duplex).

If a child element of a media element is not known or not supported, the child element SHALL be ignored and evaluated as FALSE.

NOTE 3: How PEEM determines the media type of the incoming request (i.e. in order to evaluate if a match exists for a rule containing the <media-list> condition) must be specified by the individual enabler (e.g. by additional PEL extensions and/or use of Standard PEM-1 Templates).

If present in any rule, the <service-list> element SHALL match incoming requests associated with a particular service. A <service-list> element with a list of <service> element SHALL be used to specify allowed services. A <service-list> element with an <all-services-expect> element SHALL be used to specify that all services are allowed apart from those listed as child elements. A <service-list> element with an <all-service-expect> element without any child element SHALL be used to specify that all services are allowed. The <service-list> element SHALL be evaluated to TRUE if one of its child <service> elements evaluates to TRUE. The <service-list> element SHALL also be evaluated to TRUE if all of the child <service> elements to an <all-services-except> element evaluate to FALSE. The <service-list> element SHALL be evaluated to TRUE if it contains an <all-services-except> element without any child elements.

The <service> element SHALL be used to define a service.

The attribute “enabler” SHALL specify the enabler defining the service. The “enabler” attribute SHALL be used only for Open Mobile Alliance defined Enablers. The Enabler SHALL use the OMNA registered Enabler XML schema domain token as the value of the “enabler” attribute (e.g. “poc” for the Push to talk over Cellular Enabler and “im” for the IM SIMPLE Enabler).

NOTE 4: Usage of the <service> element outside OMA can be done by extending the <service> element.

The enabler specifies how PEEM can use the information in an incoming request to recognize a request for a service. A <service> element SHALL be evaluated to TRUE if the incoming request to PEEM contains the information defined and to FALSE if not.

5.1.4.3 Combining Permissions

When evaluating policy documents based on [RFC 4745] together with the extensions described in previous sections against a URI value, the algorithm for obtaining the different rules that are applicable SHALL be as follows:

1. Those rules matching the URI value against the <anonymous-request> element SHALL take precedence over those rules based on matching it against an <identity> element. That is, if there are applicable rules based on <anonymous-request> matches, only those will be used for the evaluation of the combined permission

2. Those rules matching the URI value against the <identity> element SHALL take precedence over those rules based on matching it against an <external-list> or an <other-identity> element. That is, if there are applicable rules based on <identity> matches, only those will be used for the evaluation of the combined permission.

3. Those rules containing an <other-identity> element SHALL be used for the evaluation of the combined permission only if there are no other matching rules.

NOTE**:** The above algorithm for obtaining all the applicable rules differs from that described in [RFC 4745].

After the applicable rules have been derived based on the above algorithm, the evaluation of the combined permission SHALL be based on [RFC 4745] Section 10.2.

5.1.5 Combining Function calls and RFC 4745

OMA PEL introduces an optional PEEM ruleset language option that adds a mechanism to support function calls in rulesets modeled per [RFC 4745].

Per [RFC 4745], actions, in addition to conditions, specify all remaining types of operations the Policy Server is obliged to execute, i.e., all operations that are not of transformation type. Actions are defined by application-specific usages of this framework. The policy framework defined in this document is meant to be extensible towards specific application domains. Such an extension is accomplished by defining conditions, actions, and transformations that are specific to the desired application domain. To comply with [RFC 4745], each extension MUST define its own namespace and extensions cannot change the schema defined in [RFC 47455].

The schema for PEEM PEL ruleset language option is therefore presented in section 5.1.5.1 where changes with respect to the [RFC 4745] XML schema are highlighted in blue.

5.1.5.1 XML Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
   <xs:schema targetNamespace="urn:oma:xml:peem:ruleset:1.0"
    xmlns:gp="urn:oma:xml:peem:ruleset:1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:cr="urn:ietf:params:xml:ns:common-policy"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

    <!-- Import Common Policy-->
```

```
    <xs:import namespace="urn:ietf:params:xml:ns:common-policy"/>

    <!-- This import brings in the XML language attribute xml:lang-->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
       schemaLocation="http://www.w3.org/2001/xml.xsd"/>



    <!-- peem-ruleset//rule/actions or //rule/transformations -->
    <xs:complexType name="escapePEEM-actionType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                <xs:sequence>
<!-- The function call is modeled as as a URI that passes inputContext as argument
(input passed by value) and outputContext as populated by result. Actual binding is
implementation specific. The call is blocking for the rest of the ruleset and
processing continues as ordered with call completes with value changed in
outputContext. -->
                    <xs:element name="Functionescape" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="inputContext" type="gp:contextType"/>
                    <xs:element name="outputContext" type="gp:contextType"/>
                    <xs:element  name="FunctionCall" type="xs:anyURI"/>
                    <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>


<!-- //contextType -->
    <xs:complexType name="contextType">
        <xs:complexContent>
            <xs:restriction base="xs:anyType">
                        <xs:attribute name="value" type="xs:string"
                             use="required"/>
            </xs:restriction>
        </xs:complexContent>
```

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

5.1.5.2 Discussion

The FunctionCall can result into a function request . Errors are returned as output parameters in the output response following PEM-1 interface specification [PEEM_Callable_Policy_Interface].

In particular the call MAY launch a process defined as a business process based policy (i.e. [WSBPEL]) as described in section 5.2.

The extension is OPTIONAL and per [RFC4745] it must be ignored when not understood.

5.1.5.3 Execution model

The execution model involving a Functionescape as part of an action results into a blocking PEM1 call to a business process-based PEEM engine. ”`inputContext`”variables are passed by value. ”`outputContext`” values are used for the rest of the ruleset processing continuing to follow [RFC4745] as if these were the associated value from now on.

Returned values result into assigning the new values to the variables indentified in the outputContext sequence of attributes. Errors messages should be treated via such variables.

Returning an error is assumed as equivalent to completion of the call and therefore it unblocks the rest of the processing of [RFC4745].

5.2 PEL for Business Processes

The PEL language for Business Processes is WSBPEL 2.0 [BPEL]. Policies written using WSBPEL 2.0 SHALL conform to the WS-BPEL 2.0 schema documented in [XSD_BPEL].

This section provides a summary of the WS-BPEL syntax and constructs introduced in [BPEL]. It provides only a brief overview; the details of each language construct are normatively described in [BPEL].

5.2.1 Overview

The top-level attributes in BPEL are as follows:

- <Process>. This attribute represents a policy or a policy sub-graph (i.e. subset of combinations of conditions and actions within a policy).
- With attributes:
    - queryLanguage. This attribute specifies the query language used in the process for selection of nodes in assignment. The default value for this attribute is: "urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0", which represents the usage of [XPATH1.0] within WS-BPEL 2.0.
    - expressionLanguage. This attribute specifies the expression language used in the <process>. The default value for this attribute is: "urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0", which represents the usage of [XPATH1.0] within WS-BPEL 2.0.

    The value of the queryLanguage and expressionLanguage attributes on the <process> element are global defaults and can be overridden on specific constructs, such as <condition> of a <while> activity, as defined in [BPEL]. In addition, the queryLanguage attribute is also available for use in defining WS-BPEL <vprop:propertyAlias>es in WSDL. WS-BPEL processors MUST:

- statically determine which languages are referenced by queryLanguage or expressionLanguage attributes either in the WS-BPEL process definition itself or in any WS-BPEL property definitions in associated WSDLs and
- if any referenced language is unsupported by the WS-BPEL processor then the processor MUST reject the submitted WS-BPEL process definition.

In addition,

- suppressJoinFailure. This attribute determines whether the joinFailure fault will be suppressed for all activities in the process. The effect of the attribute at the process level can be overridden by an activity using a different value for the attribute. The default for this attribute is "no" at the process level. When this attribute is not specified for an activity, it inherits its value from its closest enclosing activity or from the <process> if no enclosing activity specifies this attribute.
- exitOnStandardFault. If the value of this attribute is set to “yes”, then the process MUST exit immediately as if an <exit> activity has been reached, when a WS-BPEL standard fault other than bpel:joinFailure is encountered. If the value of this attribute is set to “no”, then the process can handle a standard fault using a fault handler. The default value for this attribute is “no”. When this attribute is not specified on a <scope> it inherits its value from its enclosing <scope> or <process>.
- If the value of exitOnStandardFault of a <scope> or <process> is set to “yes”, then a fault handler that explicitly targets the WS-BPEL standard faults MUST NOT be used in that scope. A process definition that violates this condition MUST be detected by static analysis and MUST be rejected by a conformant implementation.
- The syntax of Abstract Process has its own distinct target namespace. Additional top-level attributes are defined for Abstract Processes.
- <documentation> construct may be added to virtually all WS-BPEL constructs as the formal way to annotate processes definition with human documentation.
- <Correlation> is defined in [BPEL].

5.2.2 Constructs

Regarding the explicit flow constructs, each business process (i.e. policy or policy sub-graph) has one main activity (or construct).

A WS-BPEL activity can be any of the following (for more details see Appendix X.1):

- <receive>: wait for a matching message to arrive.

The <receive> activity allows the business process to wait for a matching message to arrive. The <receive> activity completes when the message arrives.

- <reply>: send a message in reply to a message that was received through a <receive>.

The <reply> activity allows the business process to send a message in reply to a message that was received by an inbound message activity (IMA), that is, <receive>, <onMessage>, or <onEvent>.

- <invoke>: initiate a one-way or request-response operation offered by another resource.

The <invoke> activity allows the business process to invoke a one-way or request-response operation.

- <assign>: update the values of variables with new data.

The <assign> activity is used to update the values of variables with new data.

- <throw>: generates a fault from inside the policy processing.

The <throw> activity is used to generate a fault from inside the business process.

- <exit>: allows to immediately end a business process instance within which the <exit> activity is contained.

All currently running activities MUST be terminated as soon as possible without any termination handling, fault handling, or compensation behavior.

- <wait>: allows you to wait for a given time period or until a certain time has passed.

The <wait> activity is used to wait for a given time period or until a certain point in time has been reached. Exactly one of the expiration criteria MUST be specified.

- <empty>: insert a "no-op" instruction into a the policy evaluation or evaluation and enforcement.

The <empty> activity is a "no-op" in a business process. This is useful for synchronization of concurrent activities, for instance.

- <sequence>: define a collection of activities to be performed sequentially in lexical order.

The <sequence> activity is used to define a collection of activities to be performed sequentially in lexical order.

- <if>: Select exactly one branch of activity from a set of choices.

The <if> activity is used to select exactly one activity for execution from a set of choices.

- <while> Contained activity is repeated while a predicate holds.

The <while> activity is used to define that the child activity is to be repeated as long as the specified <condition> is true.

- <repeatUntil>: Contained activity is repeated until a predicate holds.

The <repeatUntil> activity is used to define that the child activity is to be repeated until the specified <condition> becomes true. The <condition> is tested after the child activity completes. The <repeatUntil> activity is used to execute the child activity at least once.

- <forEach>: Contained activity is performed sequentially or in parallel, controlled by a specified counter variable.

The <forEach> activity iterates its child scope activity..

- <pick>: block and wait for a suitable message to arrive or for a time-out alarm to go off, perform the associated activity.

The <pick> activity is used to wait for one of several possible messages to arrive or for a time-out to occur. When one of these triggers occurs, the associated child activity is performed. When the child activity completes then the <pick> activity completes.

- <flow>: specify one or more activities to be performed concurrently.

The <flow> activity is used to specify one or more activities to be performed concurrently. <links> can be used within a <flow> to define explicit control dependencies between nested child activities.

- <scope>: defines a nested activity with its own associated variables, fault handlers, and compensation handler.

The <scope> activity is used to define a nested activity with its own associated <partnerLinks>, <messageExchanges>, <variables>, <correlationSets>, <faultHandlers>, <compensationHandler>, <terminationHandler>, and <eventHandlers>.

- <compensate>: used to invoke functions to reverse previous operations (on all completed child scopes in default order).

The <compensate> activity is used to start compensation on all inner scopes that have already completed successfully, in default order. This activity MUST only be used from within a fault handler, another compensation handler, or a termination handler.

- <compensateScope>: used to invoke functions to reverse previous operations (on one completed child).

The <compensateScope> activity is used to start compensation on a specified inner scope that has already completed successfully. This activity MUST only be used from within a fault handler, another compensation handler, or a termination handler.

- <rethrow>: Forward a fault from inside a fault handler.

The <rethrow> activity is used to rethrow the fault that was originally caught by the immediately enclosing fault handler. The <rethrow> activity MUST only be used within a fault handler (i.e. <catch> and <catchAll> elements). This syntactic constraint MUST be statically enforced.

- <validate>: Validate format for input or output data.

The <validate> activity is used to validate the values of variables against their associated XML and WSDL data definition. The construct has a variables attribute, which points to the variables being validated.

- <extensionActivity>

- The <extensionActivity> element is used to extend WS-BPEL by introducing a new activity type. The contents of an <extensionActivity> element MUST be a single element that MUST make available WS-BPEL's standard-attributes and standard-elements.

5.2.3 Language Extensibility and support of OMA specific conditions

WS-BPEL supports extensibility by allowing namespace-qualified attributes to appear on any WS-BPEL element and by allowing elements from other namespaces to appear within WS-BPEL defined elements. This is allowed in the XML Schema specifications for WS-BPEL.

Extensions are either mandatory or optional (see [BPEL]). In the case of mandatory extensions not supported by a WS-BPEL implementation, the process definition MUST be rejected. Optional extensions not supported by a WS-BPEL implementation MUST be ignored.

In addition, WS-BPEL provides two explicit extension constructs: <extensionAssignOperation> and <extensionActivity>. See [BPEL] section 8.4. for Assignment and section 10.9. for adding new Activity Types – ExtensionActivity.

Extensions MUST NOT contradict the semantics of any element or attribute defined by the WS-BPEL specification.

Extensions are allowed in WS-BPEL constructs used in WSDL definitions, such as <partnerLinkType>, <role>, <vprop:property> and <vprop:propertyAlias>. The same syntax pattern and semantic rules for extensions of WS-BPEL constructs are applied to these extensions as well. For the WSDL definitions transitively referenced by a WS-BPEL process, extension declaration directives of this WS-BPEL process are applied to all extensions used in WS-BPEL constructs in these WSDL definitions (see [BPEL] section 14. for Extension Declarations).

The optional <documentation> construct is applicable to any WS-BPEL extensible construct. Typically, the contents of <documentation> are for human targeted annotation. Example types for those content are: plain text, HTML and XHTML. Tool-implementation specific information (e.g. the graphical layout details) should be added through elements and attributes of other namespaces, using the general WS-BPEL extensibility mechanisms.

This may be used as an extensible formalism to express OMA specific rules, e.g.(non exhaustive):

- Security rules (e.g. Authentication, authorization, GPM, confidentiality (selective), integrity, …)
- Charging rules
- Logging rules

- Privacy rules
- Preference rules
- Permission rules
- Content screening rules
  - Content categorization rules

These can be expressed in PEL as partnerlinks in WSDL. They can be predefined by OMA specifications or defined by the service provider.

5.2.4 Backward compatibility with other OMA enablers

There are no backward compatibility issues with existing enablers.

Existing OMA enabler policies are not based on [BPEL].

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-PEEM_PEL-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].

B.1 SCR for PEEM Server supporting PEL

Item	Function	Reference	Requirement
PEEM-PEL-S-001-M	Support of policy expression language	Section 5	PEEM-RSPEL-S-001 OR PEEM-BBPEL-S-001

B.2 SCR for Ruleset PEL

Item	Function	Reference	Requirement
PEEM-RSPEL-S-001-O	Support of a policy ruleset language	Section 5.1	PEEM-RSPEL-S-002-O AND PEEM-RSPEL-S-004-O AND PEEM-RSPEL-S-005-O AND PEEM-RSPEL-S-006-O AND PEEM-RSPEL-S-008-O
PEEM-RSPEL-S-002-O	Structure	Section 5.1, 5.1.2, 5.1.4.1	
PEEM-RSPEL-S-003-O	Using the <transformation> element	Section 5.1	
PEEM-RSPEL-S-004-O	Data semantics	Section 5.1.4.2	
PEEM-RSPEL-S-005-O	Extensions	Section 5.1, 5.1.3, 5.1.4	
PEEM-RSPEL-S-006-O	Combining permissions	Section 5.1.4.3	
PEEM-RSPEL-S-007-O	Function calls	Section 5.1.5, 5.1.5.1	
PEEM-RSPEL-S-008-O	XML schema	Section 5	

B.3 SCR for Business Process PEL

Item	Function	Reference	Requirement
PEEM-BPPEL-S-001-O	Support of a business process execution language	Sction 5.2	PEEM-BPPEL: OSF
PEEM-BPPEL-S-002-O	Syntax and constructs	Section 5.2, 5.2.2	
PEEM-BPPEL-S-003-O	Referenced Language	Section 5.2.1	
PEEM-BPPEL-S-004-O	Extensions	Section 5.2.3	
PEEM-BPPEL-S-005-O	XML Schema	Section 5.2	

Appendix C. Comparison between PEEM PEL needs and RFC 4745 current features (Informative)

The following constructs have been considered for the Policy Expression Language (see section 5):

1) Ruleset
2) Condition
3) Action
4) Variables (of data type integer, float, bool, string, array, struct and URI)
5) Constants (same data types as variables), plus parameterized constants
6) Operators
7) Functions

C.1 Comparisons of the policy languages to the constructs

The tables below summarizes the PEL needs against the current RFC 4745 [RFC 4745] and WS-BPEL [BPEL] specifications. In the comment column you may also see references to "work in progress" (IETF drafts with the goal of extending this policy framework with specific conditions, actions, transformations):

RFC 4745 Comparison

PEL need	**RFC 4745 support**	**Comment**
<ruleset> element	yes	Very similar, if not identical
<rule>	yes	Very similar, if not identical.
Rule complexity	Partial match, potential complete match over time	There are some potential limitations inherent in the framework model. The rule has an implicit behaviour model of supporting permissions rather than denying them (i.e. you can't write a rule of the type "if X then DENY"). FFS: Can extensions add semantics to over-write such a model, or allow multiple models, if needed.
<condition> element	yes	The framework only provides a <condition> expression container, and 3 elements. Both the 3 elements provided (identity, sphere, validity) as well as completely new <condition> elements may be needed in future to complement the current <condition> element. This is an incremental process, which can be addressed via RFC 4745 extensibility mechanism, within the work of specific application domains. This is consistent with the process described in RFC 4745, and with other documents/specifications (e.g. work-in-progress IETF drafts [PresenceAuth]

		and [WP-LOCRULES]).
multiple conditions per rule	yes	Multiple “child elements” (conditions) per rule are supported. All of the children in a rule need to evaluate to TRUE in order for the condition to evaluate to TRUE.
Complexity of a single condition	Partial match, potential complete match over time	It is difficult to assess to what extent any complex condition can be expressed, since even in the additional documents that extend the framework (e.g. [PresenceAuth] and [WP-LOCRULES], the extensions do not include complex expressions using logical and/or mathematical operators. This does not mean that extensions to support complex expressions are not possible, just that they are not yet readily available.
<action> element	yes	Very similar, if not identical
multiple actions per rule	yes	Multiple “child elements” (actions) per rule are supported. There is an implied “permissions” combining algorithm. FFS: whether the permissions combining algorithm has limitations, that may be desirable to be changed, over-riden or removed , if need be.
Complexity of a single action	Partial match, potential for complete match over time	It is difficult to assess to what extent any complex action can be expressed, since even in the additional documents that extend the framework (e.g. [PresenceAuth] and [WP-LOCRULES], the extensions do not include complex expressions using logical and/or mathematical operators. This does not mean that extensions to support complex expressions are not possible, just that they are not yet readily available. In particular, a limitation that is acknowledged in RFC 4745 is the lack of support of actions that may need external support (see more details on the “functions” requirement).
Variables	Partial match, potential for complete match over time	Variables are introduced as part of the introduction of conditions/actions. The data types may be implicit (but most examples available show the use of strings, Boolean, integer). However, it is likely that new conditions/actions can add variables of any data type (there is no evident restrictions, just lack of a clear

		statement in that sense).
Constants	Partial match, potential for complete match over time	Same as above
Parameterized constants	Yes	Based on the extension of the RFC 4745 defined for PEEM, see Section 5.3, any function can be defined.
Operators	Difficult to assess, no examples	The framework does not explicitly state support for operators (logical or mathematical) to be used in expressions, simply because in the framework, and in the other "work-in-progress", the expressions are reduced to 1 variable. If a new application will require complex conditions and/or actions, than at that time operators would have to be supported as well.
Functions	Yes	Based on the extension of the RFC 4745 defined for PEEM, see Section 5.3, any function can be defined.
<transformation> element does not have an equivalent in PEEM	Exists in [RFC 4745]	The framework describes how <action> and <transformation> are to be used, but does not provide any child elements for those. It leaves those for other specifications to be added as extensions. Examples provided in [PresenceAuth] use <action> but no <transformation>. Examples provided in [WP-LOCRULES] use <transformation> but do not use <action>. FFS: whether both <action> and <transformation> are needed by OMA application domains and/or other resources, or whether the <action> element semantics could be extended in future to include whatever semantics <transformation> carries – for simplification reasons.
Matching input parameters (via PEM-1) to policy parameter	Not addressedOut of scope for RFC 4745	Out-of-scope for RFC 4745.

WS-BPEL Comparison

PEL need	**WS BPEL support**	**Comment**
<ruleset> element	yes	Workflow construct with same function
<rule>	yes	Workflow construct with same function
Rule complexity	No Limitations	The WS-BPEL language has no limitations regarding the complexity of the rules.
<condition> element	yes	The WS-BPEL language has no

		limitations regarding the use of conditional elements in the rules.
multiple conditions per rule	yes	Multiple “child elements” (conditions) per rule are supported.
Complexity of a single condition	No Limitations	The WS-BPEL language has no limitations regarding the complexity of the rules.
<action> element	yes	Very similar, if not identical
multiple actions per rule	yes	Multiple “child elements” (actions) per rule are supported. There is an implied “permissions” combining algorithm.
Complexity of a single action	No Limitations	The WS-BPEL language has no limitations regarding the complexity of the actions.
Variables	yes	Variables are available in WS-BPEL
Constants	yes	Constants are definable in WS-BPEL
Parameterized constants	yes	This concept is available as WS-BPEL can obtain data from any type of source during the execution of the work-flow, e.g. based on input attributes like application ID.
Operators	yes	WS-BPEL support the standard operators <, >, = etc.
Functions	yes	WS-BPEL support calling other work flows.
<transformation> element does not have an equivalent in PEEM	yes	The framework for WS-BPEL supports working on data based on the work flow handling any type of data transformation.